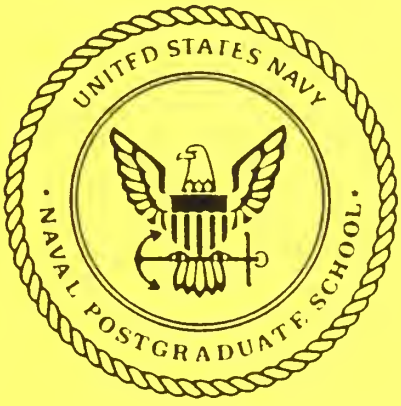
NPSOR-93-012

NAVAL POSTGRADUATE SCHOOL

Monterey, California



RANGE CALIBRATION STUDIES
INTERIM REPORT

Colin R. Cooper
Robert R. Read

May 1993

Approved for public release; distribution is unlimited.

Prepared for:
Naval Undersea Warfare Engineering Station
Keyport, WA 98345

NAVAL POSTGRADUATE SCHOOL
MONTEREY, CALIFORNIA

Rear Admiral T. A. Mercer
Superintendent

Harrison Shull
Provost

This report was prepared in conjunction with research partially funded by Naval Undersea Warfare Engineering Station, Keyport, Washington.

This report was prepared by:

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION /AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) NPSOR-93-012	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Naval Postgraduate School 6b. OFFICE SYMBOL (*If applicable*) OR/Re	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (*City, State, and ZIP Code*) Monterey, CA 93943	7b. ADDRESS (*City, State, and ZIP Code*)
8a. NAME OF FUNDING/SPONSORING ORGANIZATION NUWES 8b. OFFICE SYMBOL (*If applicable*)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (*City, State, and ZIP Code*) Keyport, WA 98345	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION

11. TITLE (*Include Security Classification*)
Range Calibration Studies — Interim Report

12. PERSONAL AUTHOR(S)
Colin R. Cooper and Robert R. Read

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (*Year, month day*)	15. PAGE COUNT
Technical	FROM_____TO_____	1993 May	77

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (*Continue on reverse if necessary and identify by block number*)
FIELD	GROUP	SUB-GROUP	

19. ABSTRACT (*Continue on reverse if necessary and identify by block number*)

This report describes the work performed on the Range Calibration project for calendar year 1992. The main efforts were in the areas of data retrieval and staging, and the development and application of some error type diagnostic techniques. Study of the use of the diagnostics showed usefulness but also revealed a number of lacunas, which are described. Recommendations are made for continuation work.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT [x] UNCLASSIFIED/UNLIMITED [ ] SAME AS RPT. [ ] DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL Robert R. Read	22b. TELEPHONE (*Include Area Code*) (408) 656-2382 22c. OFFICE SYMBOL OR/Re

DD Form 1473, JUN 86 *Previous editions are obsolete.* SECURITY CLASSIFICATION OF THIS PAGE
S/N 0102-LF-014-6603 UNCLASSIFIED

RANGE CALIBRATION STUDIES
INTERIM REPORT

C. Cooper and R. Read

Abstract

This report describes the work performed on the Range Calibration project for calendar year 1992. The main efforts were in the areas of data retrieval and staging, and the development and application of some error type diagnostic techniques. Study of the use of the diagnostics showed usefulness but also revealed a number of lacunas, which are described. Recommendations are made for continuation work.

Introduction

The report describes the work performed on the Range Calibration project during calendar year 1992. It is an interim report. Much was learned and those things are described. Documentation is made in the appendices of the computer programs that were written. The presentation style presumes that the reader is familiar with the setting, the terminology, and the previous work; see references [9, 10, 11, 12].

The goal of the research is two-fold:

i. To establish the relative positions and orientations of the several short baseline tracking arrays.

ii. To devise ray tracing methods that will reduce the separation of two versions of track to acceptable levels and in a way that is consistent throughout the range. Apparent discontinuities, anywhere, are to be removed.

The two goals interact with one another as will be seen. Two or more versions (from different arrays) of the track of a single vehicle for specialized time intervals show average discrepancies of 15 to 50 feet with considerable frequency. Yet the range survey methods are designed to limit these discrepancies

to just a few feet, [1]. A number of interpretations of this contradiction have been offered:

a. Shortcomings of the ray tracing initialization methods.
b. Shortcomings of the ray tracing algorithm itself.
c. Inappropriate extrapolation of the depth-velocity (DV) profile; (also called the water column).
d. Spatial variability of the water column.
e. Faulty timing synchronization of the target vehicle "pinger" and the range clock.
f. Other miscellaneous sources, e.g. under-water currents.

We would like to suggest that the "Hurst effect" [2, 6] be considered also. This effect is associated with a component of variability that changes very slowly in time. This may be a small size but large scale effect (i.e., affecting a large volume, perhaps the entire range) that biases the measurements taken over a comparatively small time interval. Such could affect the array survey procedure which lasts but a few hours. The presence of a "Hurst effect" could be established (or refuted) if we were to repeat the survey of an array, say monthly, for a year or so and compare the results.

Previous work has considered some of the items listed above. The importance of proper initialization of the elevation angle for ray tracing has been established in [10]. An improved technique was suggested and has been adopted. There may be room for further improvement however. The same reference drew attention to some weaknesses in the tilt adjustment method, and the correct technique for that has also been implemented. Two ray tracing methods, isospeed and isogradient, were compared in [10]. Seldom is the discrepancy more than a foot or so.

The extrapolation of the DV values must be done to an additional 75 feet. Current practice is to use constant value extrapolation to the deeper water layers.

Some graphs of the water columns leave one to doubt the validity of this method and, as shown in [10], work with an ad hoc but graphically pleasing extrapolation suggests that errors from this source can be 10 feet or more in size. The recommendation was to measure the water column to the deeper levels. It appears that this idea has been rejected because of i) fear of damaging the instrument, and ii) the requirement for more frequent cleaning of the instrument. There is much murk in the deeper layers near the bottom. Thus, there is a need for a usable extrapolation method.

The question of spatial variability, i.e., changes in the water column as a function of horizontal position, has never been treated. Indeed a thorough treatment would require a rather considerable resource commitment. However it is now feasible to experiment with some rather simple models because of the recent advances in computing techniques and machinery. Some suggestions will be made later.

A timing synchronization error model was constructed [11] and tested with real data. The method will reduce the track separation in the double overlap regions, but not in a way that is consistent from region to region. This interpretation of the separations of track must be rejected, at least at the scales currently experienced. It could have value as a data smoothing device and can be held in reserve for possible use after other sources of systematic error have been reduced.

Other sources of error have been mentioned elsewhere [3, 4, 7, 8, 13], but generally their magnitudes are small, certainly not enough to account for the separations that we observe regularly.

Recent Work

Most recently we have been pursuing an idea due to J. Knudsen. It is designed to exploit the occasional appearance of three versions of track that occur in the triple overlap regions (TORs), see Figures 1 and 2. Since three arrays are receiving transit time information at a common set of time epochs (point counts), Knudsen's idea is to use long baseline position location methods for purposes of establishing "true" poslocs which can be used as a reference for adjusting the parameters of the arrays and the elevation angle that initializes the ray tracing. The three-dimensional long baseline algorithm, called LONGBASE, has been developed, programmed, and tested [9]. We proceed to describe our experiences when it is put to use.

The initial use was made by Gembarski in his master's thesis [5]. It seems clear that if the locations of the three c-phones of the contributing arrays of a given TOR are known, and there is no other source of systematic error, then the LONGBASE algorithm will supply the correct elevation angles at each c-phone. Of course these are the angles needed to initiate the ray tracing. These angles are valid for their respective arrays. The azimuthal directions from those arrays must conform to the LONGBASE posloc[1].

Consider a single array, i.e., an interior one in Figure 2, where by interior array, we mean one that has a full complement of six TORs. The LONGBASE algorithm can be used to provide corrected elevation angles in all six directions. These can be used to adjust the XTILT and YTILT orientation angles of the interior array. Only two degrees of freedom are consumed in so doing as we have assumed no other source of systematic error. This suggests that the

---

[1] posloc is brief for position location.

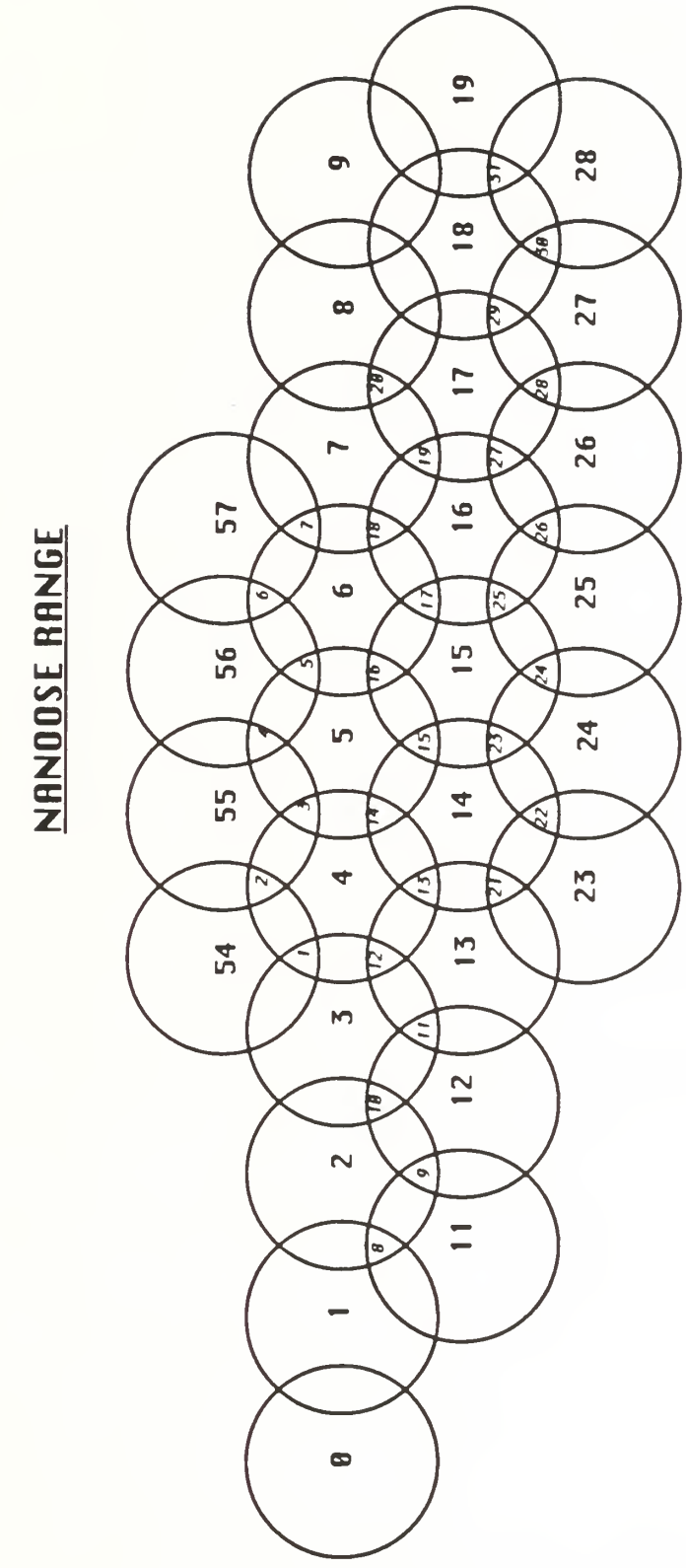


Figure 1. Plan View of Nanoose Range Showing TORs

remaining degrees of freedom might be used to adjust other parameters that could be out of kilter.

Gembarski looked into this question through a simulation study. He placed target vehicles at known locations in each of the six TORs of an interior array. Assuming known locations of the interior and all satellite arrays, the RAYFIT algorithm was employed to generate transit times to all arrays from all target locations. This done, then the parameters of the interior array were changed somewhat to fictitious but assumed official values, and using these values, the short baselines posloc was developed in each of the six TORs. Of course such poslocs were at variance with those produced by the satellite arrays. Gembarski undertook to find a method, using LONGBASE, by which the true parameters of the interior array could be recovered.

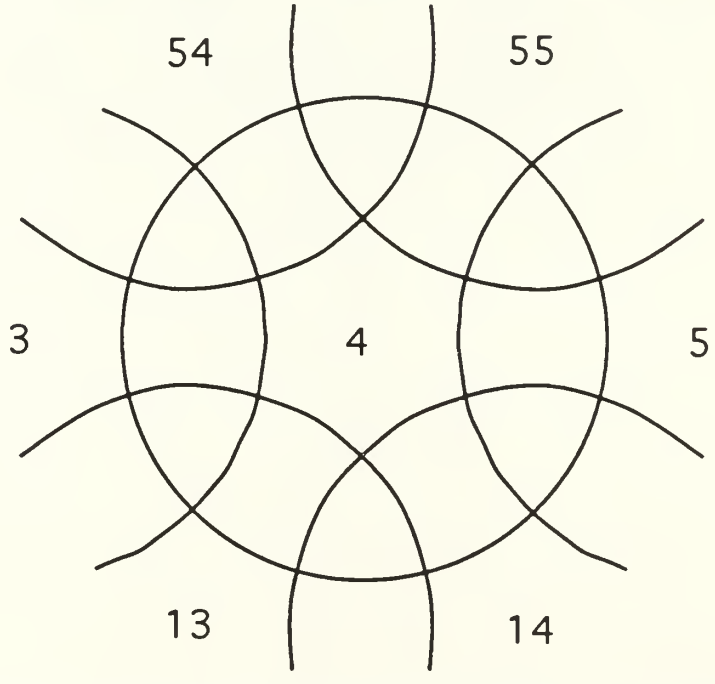


Figure 2. Interior Array with six TORs (exaggerated)

The method was successful but slow. It involved an iteration search scheme on each of the six coordinates (three for location and three for orientation) one by one. Gembarski also experimented using the case of errors in the satellite array parameters as well, (but of smaller magnitude than the errors in the interior array parameters). This too was successful.

Since then we have been trying to apply similar methods to real data from the Nanoose range. This is a quantum step beyond the simulations by Gembarski for a number of reasons:

i. Using real data, we do not know the real poslocs of the target vehicle.
ii. All arrays must be assumed to have their parameters in error, and there is no reliable way to choose an interior array that is more "out of kilter" than the others.
iii. Real data do not provide the stability that comes with simulated data which is free of other sources of systematic error.
iv. Adjustments to the parameters of a single array will not produce concurrence. Nor will follow-on adjustments to the parameters of neighboring arrays. There is considerable likelihood of endless oscillation, back and forth among several arrays, without reason to expect stabilization.[2]

Our initial efforts showed that a single coordinate search was not going to be successful. Study of the behavior of the computations has led us to seek some diagnostic aids that allow prioritization of the various possible error sources. This is to be followed by a series of small measured adjustments until further change leads to a new type of adjustment identified by the priority scheme. The execution of this phase has yet to begin. The details of the diagnostic aids are not complete. The remainder of the report will describe the progress and document the computer programs produced thus far.

---

[2]Such was an earlier experience [12].

Immediate Background

Figure 1 shows the plan view of the Nanoose range and the arrays by number. Also included is the numbering system we have assigned to the TORs. Further there may be occasion to refer to some of the DORs (double overlap regions), but we have not yet found a need for an identification system for these.

Any array designated for immediate parameter adjustment will be called a base array. Since every array shares some overlap regions with its neighbors, these immediate neighbors to the base array will be called satellites. An interior array is one that shares TORs with each of a full complement of six satellite arrays. Thus the interior arrays are subject to the maximal possible amount of potentially conflicting mismatches. It seems that they should be adjusted first. The non-interior arrays are influenced by a fewer number of data in TORs and they will be treated afterwards.

The term "adjusting an array" refers to the revision of its six parameters, three for location and three for orientation. It will also include the possible revision of the elevation angle determination that initializes the ray tracing algorithm. It is hoped that any such revision would be common for all arrays in the range, but we remain alert to the possibility that individual arrays may have properties that call for individualized elevation angle determination rules. Also it may occur that the elevation angle determination rules may change with the peculiarities of the water column. Thus there are up to seven degrees of freedom involved in describing the adjustment of an array.

Figure 3 contains a different view of the range. The circles of array influence (domains of tracking) are marked lightly in the background. The bold lines connect the center of each interior array to points selected from each of its TORs.

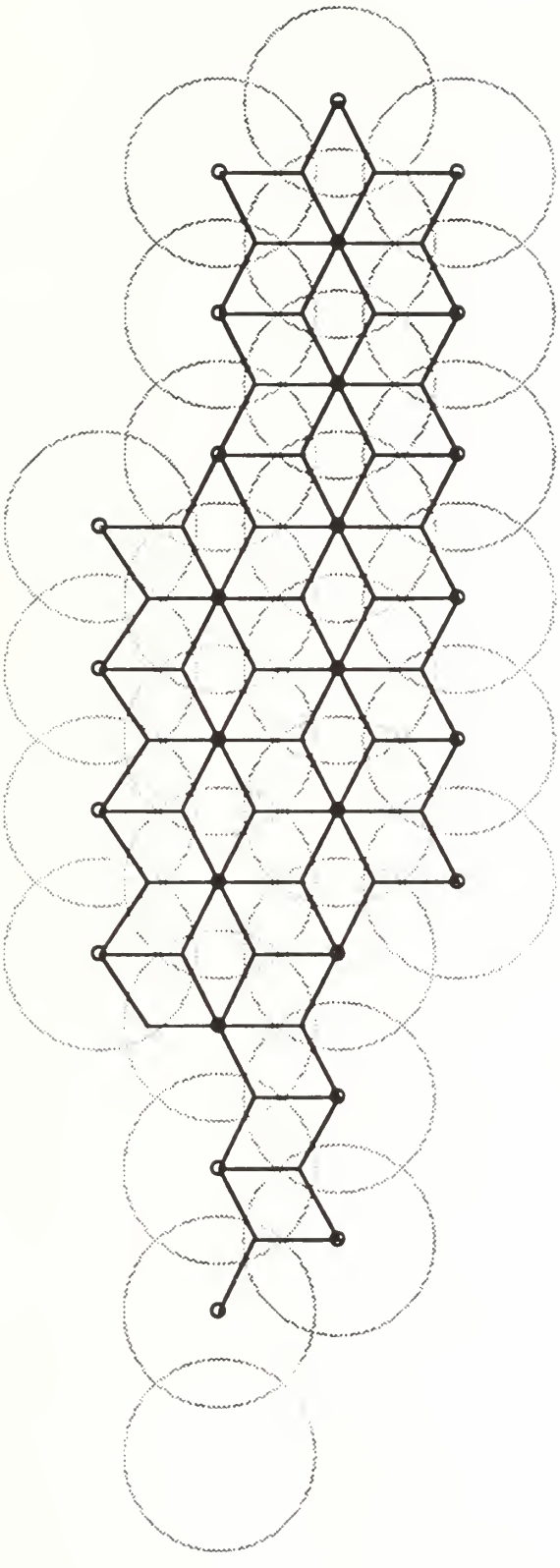


Figure 3. Diagram Showing Lines Connecting Array Centers with TORs

The images here have an effect of making the researcher over-confident. After all the LONGBASE algorithm produces poslocs without making use of any array orientation (tilt or rotation) information. Thus it seems that the array locations can be found separately, at least relative to each other. This is rather naive however. One must keep in mind that the range is three-dimensional; the various arrays are at differing depths and the poslocs indicated by the lines intersecting in the TORs are much shallower, but certainly not necessarily at common depths. The true image behind Figure 3 is more likened to that of an egg carton, and the troughs and peaks are variable.

With this imagery one views the figure as possessing greater elasticity. The bold connecting arms do not have fixed length because the speed of sound in water is not constant. Generally the deeper the c-phone, the greater the length because sound speed increases with depth. Still it may be possible to tune the relative locations of the arrays using longbase methods provided i) the idealization of no other source of systematic error is tenable, and ii) the depth of the target is stable[3].

Basis of Diagnostics

Errors in the seven array parameters lead to errors in the poslocs. The nature of the errors in the poslocs should provide discriminating information concerning the contributing roles of the array parameters. There are some interacting effects however. This section is devoted to describing the relationships between the errors in the poslocs and the appropriate parameter adjustments that would reduce them.

---

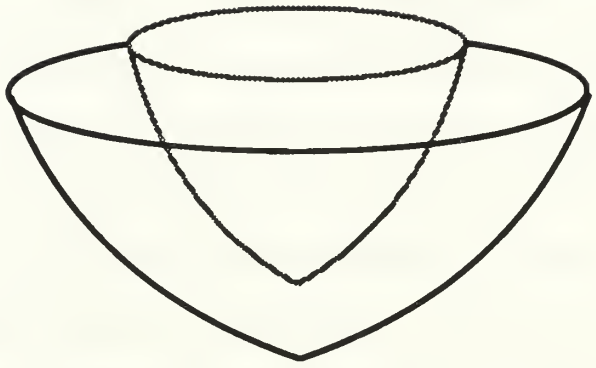
[3]Asynchronous tracking in which the depth is determined by a separate data source has been mentioned by L. Anderson.

In real situations we do not know the errors in the poslocs. We only have information about discrepancies between versions of track produced by different arrays. Such discrepancies must serve as surrogates for posloc errors. There is certainly a correlation between the two and it will be necessary to identify those discrepancies that are best correlated with the posloc errors. Presumably such correlations will become stronger as the parameter errors are reduced.

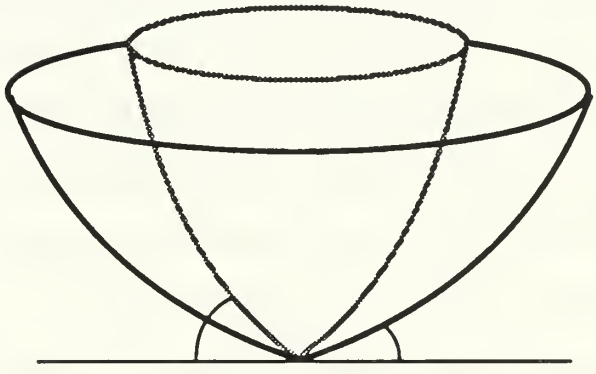
For the moment however, it is assumed that the errors in poslocs are known. Indeed we go a step further and assume that a true circular track is available. It is at a fixed depth and centered at the horizontal coordinates of the base array's c-phone. (Later it will be replaced by posloc information from the satellite arrays in the directions of the TORs.) It is convenient to assume the circular true track for purposes of exposition. We visualize the errors produced by comparing the track produced by the base array with that of the true track.

To begin, let us suppose that the only array parameter error is in the vertical position of the base array's c-phone. The situation is depicted in Figure 4a. The light inner circle represents the true track and the bold outer circle is the base array's measured track. The curved lines are the ray traces and the figure may be thought of as two half coconut shells, one inside the other. Here the true depth of the c-phone is less than that used to produce the measured track. The speed of sound in water increases with depth and, for this reason, the true track is shallower and the horizontal range (radius of the circle) is smaller.

Figure 4b shows the same kind of error structure but for a different reason. This time the depth of the c-phone is correct but the elevation angle that initializes ray tracing is in error. The true elevation angle is larger than the constructed one. In this case the ray will leave the fast water layers sooner and be stopped at a smaller radius at the more shallow depth.



a) Error is in the depth of the c-phone



b) Error is in the ray elevation angle

Figure 4. Perspective View of Horizontal Circular Track.

The major point in Figure 4 is that the two kinds of parameter error produce the same kind of posloc error. This means that vertical location adjustment and elevation angle adjustments must be treated jointly. They can mask one another. Of course if the shells in Figure 4 are reversed then the directions of the parameter adjustments are also reversed.

Next let us study the case in which the only array parameter error is that of a horizontal displacement in its locations. The situation is depicted in Figure 5, which shows a plan view of the two circles. The true track is in light and the measured one is in bold. If one were to plot the (polar coordinate) posloc error curve (measured radius less reference horizontal range from a common point)

one would get a periodic function of azimuth. It could be fitted with a sine wave and the phase angle of that wave would correspond to the direction of the dashed line in Figure 5. This theme will be developed later in the report. For now it suffices to say that the two horizontal displacement parameters can be treated jointly.

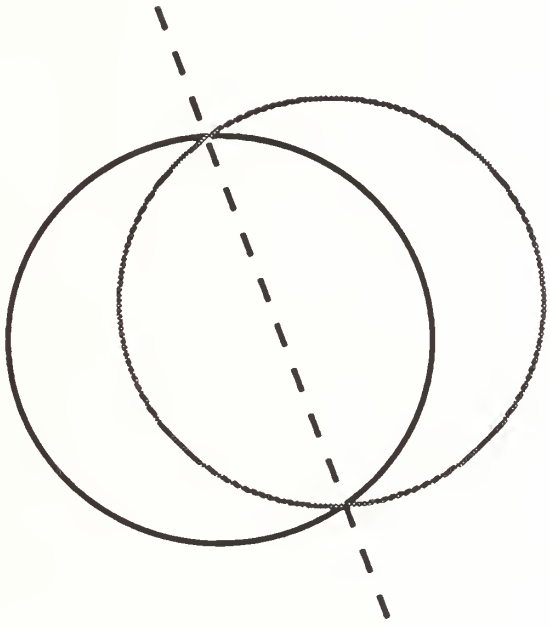
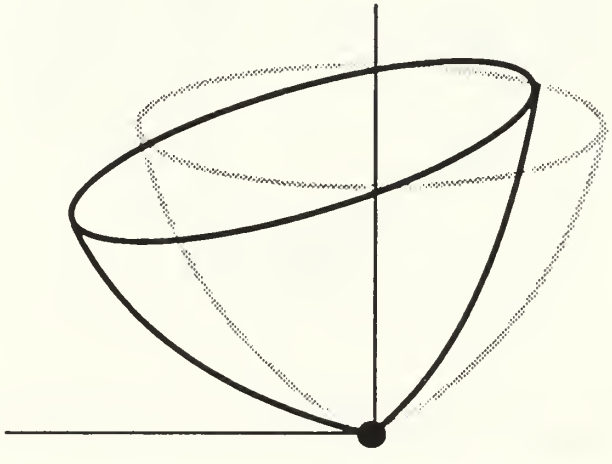


Figure 5. Circular Track with Horizontal Array Displacement

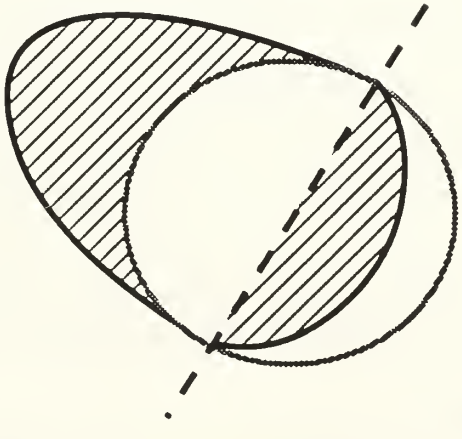
Next let us consider the effect of an error in azimuth in the ZROT angle. If this is the only error then our two circles will be congruent, but of course poslocs coupled by point counts will be shifted out of alignment. If a ZROT error occurs in conjunction with those in Figure 4, then the error wires connecting the two circles will be given a cant. If a ZROT error is included in the conditions surrounding Figure 5, then the circles will remain the same but the pairing of points on them will change.

Finally let us examine what happens to our horizontal circle track when the only errors are in the tilt angles, XTILT and YTILT. The situation is depicted in

Figure 6, with the coconut shell imagery and also a top view. Clearly the two tilt angles should be treated jointly and the figure indicates that an axis of tilt should be located for use in corrective action. The more important aspect of this figure is that the original circle, when tilted, is no longer a circle but becomes egg-shaped. Moreover it is not necessarily a planar figure. The original plane may become warped. The reason for these changes is again due to the gradient of the speed of sound with water depth. Thus a plot of the vertical (horizontal) errors as a function of azimuth would be a periodic function with deeper negative lobes than positive ones.



a) Perspective View



b) Plan View

Figure 6. Effect of Tilted Array on Circular Track

Implementation of Diagnostics

In the previous section we presented the effects of errors in the seven array parameters and identified subsets of parameters that should be treated jointly. Let us put this to use.

We cannot expect to obtain a circle of track about a base array. It does seem reasonable to expect track poslocs (or even segments of track) in each of the TORs. If the base array is an interior array then the data from the six TORs will provide poslocs of a target vehicle in six roughly equally spaced azimuthal directions and roughly at a common horizontal range.

The true poslocs will not be available but the satellite arrays will provide some posloc information that can be used for comparison. The LONGBASE posloc can also be computed. One must continually keep in mind the fact that these comparison poslocs are generated from arrays that also are subject to error in their parameters. Presumably these comparison poslocs will have some internal coherence. If so then there will be some sort of consensus point (or region) and we want the reader to allow us to refer to such a point even though it is yet to be defined explicitly.

Poslocs produced by the base array using the short baseline ray tracing algorithm will be called the measured posloc and the designated consensus value in each TOR will be called the reference posloc. The posloc errors will be defined by

$$\text{error} = \text{measured} - \text{reference}$$

and they can be computed either in the three Cartesian coordinates or in terms of horizontal, vertical, and azimuth.

Let us ignore azimuth for the moment and contemplate plots of the horizontal and vertical components of the measured and the reference for all six TORs in the

(h, z) plane. Connect the matched pairs with straight line segments and look for some sort of consistency. If the condition depicted in Figure 4 is dominant, then the six line segments will roughly coincide. For definiteness assume the measureds are beneath the reference and the slopes are about the same. Hence, we should be jointly concerned with the depth of the base array c-phones and the elevation angles for ray tracing. How might we exploit this diagnosis?

Let z_c be the variable representing the depth of the c-phone and let θ_i be the elevation angle in the i^{th} of the six azimuthal directions. Consider a first order-bivariate Taylor series expansion of the horizontal and vertical errors.

$$\begin{aligned} \Delta h_i &= \frac{\partial h}{\partial z_c}\Delta z_c + \frac{\partial h}{\partial \theta_i}\Delta\theta_i \\ \Delta z_i &= \frac{\partial z}{\partial z_c}\Delta z_c + \frac{\partial z}{\partial \theta_i}\Delta\theta_i \end{aligned} \tag{1}$$

for $i = 1, \ldots, 6$. This is a system of 12 equations in 7 unknowns $(\Delta z_c, \Delta\theta_i, \ldots, \Delta\theta_6)$. With more equations than unknowns we will employ the least squares compromise solution. But before this can be done, there is still the question of determining the partial derivative coefficients. This is a difficult task because they are functions of the water column (DV profile) as used by the ray trace algorithm. At the moment we do not have a reasonable way to manage this problem.

Suppose we are blessed with a solution to the above system. The value Δz_c is the estimated correction to the depth of the c-phone. It may not be significant, but if it is we may choose not to use all of it. Don't forget that the reference values are also subject to error and it may be unwise to treat them otherwise. It will be useful to recompute the measured poslocs after the change and study the effect.

The six values $\Delta\theta_1, \ldots, \Delta\theta_6$ may contain some useful information. They could present a smooth coherent wave when plotted as a function of azimuth. Such a

plot could suggest either a constant value $\Delta\theta$ error, or some tilt error information or both. If a sine wave were fitted to these numbers, then the phase angle would suggest the direction of the tilt error axis indicated in Figure 6. Again, slow steady corrective action seems advisable.

Next suppose that the vertical errors are judged not significant and the horizontal errors exhibit a smooth periodic curve when plotted against azimuth. We liken this condition with Figure 5. A sine wave fit to these pairs, if significant, can be used to identify the direction and size of the horizontal displacement. If the size of the displacement is negligible, then perhaps the amplitude of the wave can be used to locate and correct some tilt errors.

If both horizontal displacement and tilt errors are present in the data, then the periodic function representing Δh versus azimuth could have two cycles, i.e., superimposed sine waves with differing phase angles and amplitudes. More than six data points would be required to separate the two waves.

Initial Experiences

Some TOR data was extracted from the general files that contain the transit times. It was necessary to make a number of ad hoc decisions in order to reduce these data to usable form.

First of all, a single TOR track segment was defined in terms of the existence of three versions of track. It turned out that this technique produced track segments whose horizontal ranges from their parenting arrays were not particularly close (as had been presumed) but quite variable. The condition is a common one. Because of it, liberal use was made of relative error in place of signed magnitude of error.

rel error = (measured– reference)/reference[4]

The lengths of the track segments in the TORs were also quite variable, ranging from as few as ten point counts (time epochs) to sometimes over one hundred. It seemed wasteful to apply the LONGBASE program to all of them and it was decided to filter each of the three versions of track. More specifically the central eleven point counts were chosen; the transit times were converted to poslocs; the eleven poslocs were averaged; and the resulting smoothed locations (for each of the three contributing arrays) were assigned to the central (sixth) point count.

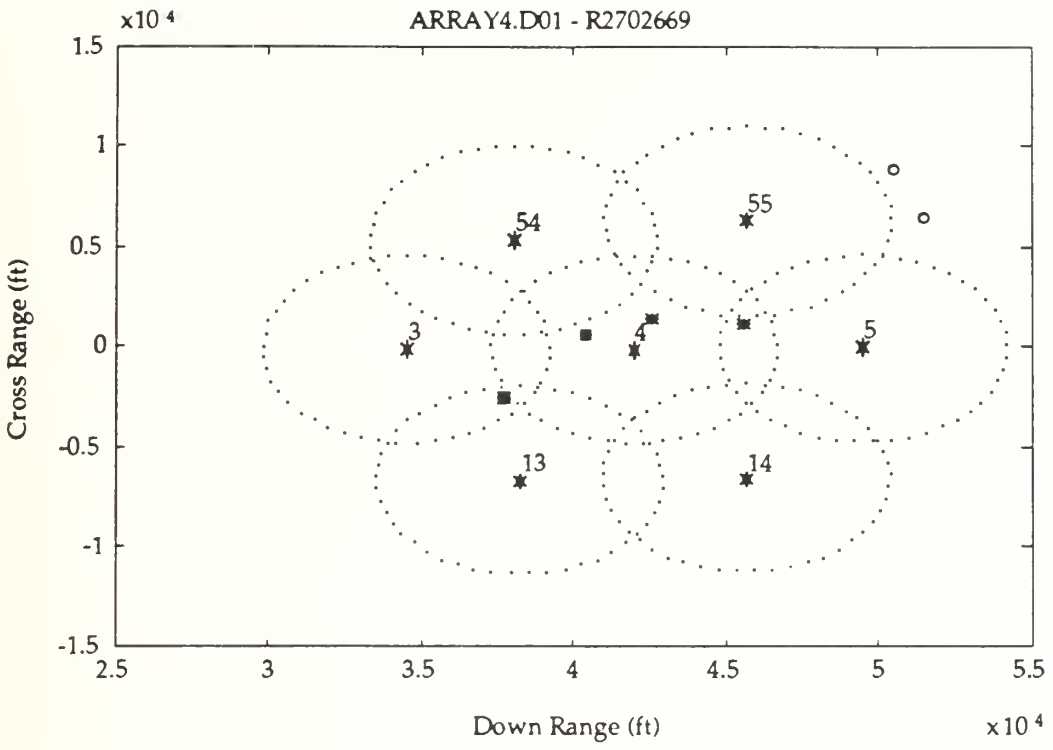
The next step was to apply the program RAYFIT to each of the three smoothed poslocs and thereby produce the individual c-phone transit times. It is these three transit times that were input to the LONGBASE program. The output will be called the LB posloc.

There were several instances in which the LONGBASE program did not converge. That is, the iterative technique that seeks the intersection of the three wave fronts was marching off to a place above the surface of the water. Such experience certainly supplies evidence that at least one of the arrays is off the mark. It also supplies the unsettling evidence that the LB posloc may not serve as well as a reference posloc. It is highly sensitive to the locations of the three c-phones.

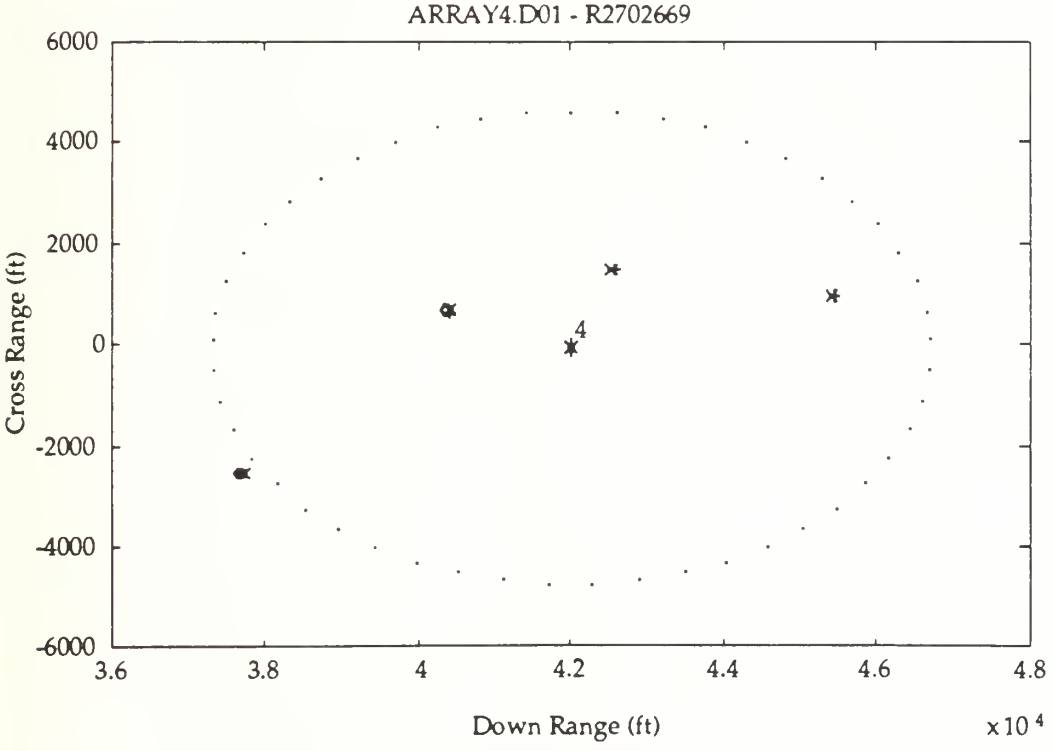
Figure 7 shows a plan view of four of the TOR LB poslocs in an instance in which array 4 is the base array. The reader should note the disparity in the three horizontal ranges for each. This type of condition was rather common and, as mentioned before, led to the use of relative error in place of error. Unfortunately

---

[4]When dealing with the vertical distances, the denominator is the distance of the reference vertical from the depth of the c-phone.



a) Low Resolution



b) High Resolution

Figure 7. Location of 3 Array Input Filtered POSLOCs

it is possible that relative error may be a non-trivial function of the reference variable, e.g., horizontal range or elevation angle. (This issue is yet to be studied.)

At this point we display some figures that illustrate the kinds of information we get using real data. Some data sets were selected and marked

2667_4 2670_4 1884_4

2672_16 1884_16

The first four digits identify the track and day. The number to the right of the lower bar identifies the base array. Arrays 4 and 16 are separate of one another in the sense that neither is a direct satellite of the other. The data that supports the figures appears in Table 1.

The first sets are horizontal range vs. depth plots, or (h, z) plots, and all are collected in Figure 8. The scales are quite deceptive in that the horizontal spacings, in one case, represent 500 feet and the vertical spacings are 20 feet. Like discrepancies occur in the other cases. The slopes of these line segments typically run from -1 to -5, i.e. they represent dz/dh.

a2669_4.out						
Az (rad)	18.7°	74.1°	154.0°	209.5°		
h (ft)	3756.4	1605.0	1798.8	4991.1		
Δh (ft)	35.5	51.6	26.9	7.6		
z (ft)	396.9	347.5	345.8	358.8		
Δz (ft)	127.8	79.6	47.5	35.6		
dz/dh	-3.6004	-1.5439	-1.7655	-4.6762		
θ (rad)	0.2674	0.5914	0.5188	0.1876		
Δθ (rad)	0.0147	0.0416	0.0168	-0.0203		
Rel. h	0.0095	0.0332	0.0152	0.0015		
Rel. z	0.4749	0.2971	0.1592	0.1101		
Rel. θ	0.0580	0.0758	0.0334	-0.0976		

Table 1

a2670_4.out						
Az (rad)	8.2°	167.4°	194.1°	270.1°		
h (ft)	3683.6	3906.7	3922.2	2611.0		
Δh (ft)	10.1	15.6	16.8	28.7		
z (ft)	388.8	393.5	370.5	359.3		
Δz (ft)	37.7	60.8	63.7	72.8		
dz/dh	-3.715	-3.8887	-3.7894	-2.5407		
θ (rad)	0.2511	0.2413	0.2464	0.3753		
Δθ (rad)	-0.0066	-0.0021	-0.0021	0.0156		
Rel. h	0.0028	0.004	0.0043	0.0111		
Rel. z	0.1074	0.1827	0.2076	0.2541		
Rel. θ	-0.0255	-0.0086	-0.0086	0.0432		

a1884_4.out						
Az (rad)	23.9°	84.3°	154.0°	210.9°	273.0°	315.9°
h (ft)	4060.3	1813.7	4171.7	4285.4	2263.2	3346.0
Δh (ft)	7.6	6.4	8.6	18.2	24.8	12.2
z (ft)	401.7	368.6	379.3	387.1	368.1	373
Δz (ft)	31.1	11.9	35.2	75	55.9	40.4
dz/dh	-4.0771	-1.8696	-4.0873	-4.1252	-2.2519	-3.3082
θ (rad)	0.2199	0.4845	0.2196	0.2209	0.4168	0.2792
Δθ (rad)	-0.0174	-0.0043	-0.0168	-0.0086	0.0113	-0.0080
Rel. h	0.0019	0.0035	0.0021	0.0043	0.0111	0.0037
Rel. z	0.0839	0.0334	0.1023	0.2403	0.1791	0.1215
Rel. θ	-0.0732	-0.0088	-0.0711	-0.0376	0.0279	-0.0280

Table 1 (continued)

a2672_16.out						
Az (rad)	27.1°	85.1°	264.2°	316.0°		
h (ft)	3707.6	2985.6	3999.6	5851.1		
Δh (ft)	2.9	-18.6	-11.8	15.5		
z (ft)	279	81.3	303.3	471.7		
Δz (ft)	10	-45.4	-45.4	91.7		
dz/dh	-3.4037	-2.4348	-3.8455	-5.9020		
θ (rad)	0.2676	0.3699	0.2287	0.1483		
Δθ (rad)	-0.0173	-0.0243	-0.0319	-0.0102		
Rel. h	0.0008	-0.0062	-0.0029	0.0027		
Rel. z	0.0372	-0.3583	-0.1302	0.2413		
Rel. θ	-0.0606	-0.0616	-0.1225	-0.0641		

a1884_16.out						
Az (rad)	16.3°	83.5°	257.8°	351.5°		
h (ft)	3676.6	5450.4	4620.1	5594.9		
Δh (ft)	-22.1	-18.9	13.9	-36.4		
z (ft)	181.9	219.2	471.7	165.5		
Δz (ft)	-70.9	-89.8	67.2	-174.2		
dz/dh	-3.2089	-4.7453	-4.8474	-4.7822		
θ (rad)	0.2698	0.1680	0.1838	0.1574		
Δθ (rad)	-0.0434	-0.0481	-0.0117	-0.0669		
Rel. h	-0.0060	-0.0035	0.0030	-0.0065		
Rel. z	-0.2805	-0.2906	0.1661	-0.5128		
Rel. θ	-0.1386	-0.2224	-0.0598	-0.2984		

Table 1 (continued)

The asterisk marks the LB posloc generation as described earlier from the transit times of the three filtered poslocs. The small circle marks that member of the filtered poslocs that is associated with the base array. This is the measured posloc.

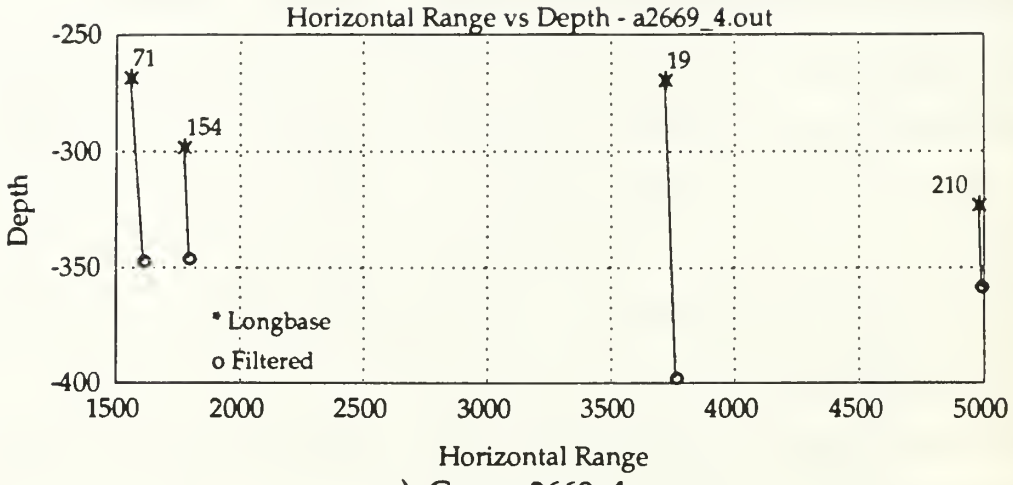
Cases a, b, c of Figure 8 all have the reference posloc shallower than the measured one. Array 4 is the base array in each case. These graphs consistently support the presence of errors in z_c, the depth of the c-phone, or in the elevation angles, or both. The corrective action can be approximated by use of the system of equations (1). The high variability of the range values tells us that the condition described in Figure 7 is present. We do not have roughly common reference horizontal ranges.

Cases d, e of Figure 8 both have Array 16 as the base. This time we do not get consistency in the vertical placement of reference and measured poslocs. But there is a modicum of consistency in the two cases for common azimuth direction. The LB posloc is below the measured one at the 16° and 27° azimuths; it is above at the 258° and 264° azimuths; it is below again at the 316° and 351° azimuths. But the signal does not seem to continue at the 82° and 85° azimuths. There may be a tilt error condition here. It would be useful to have more data.

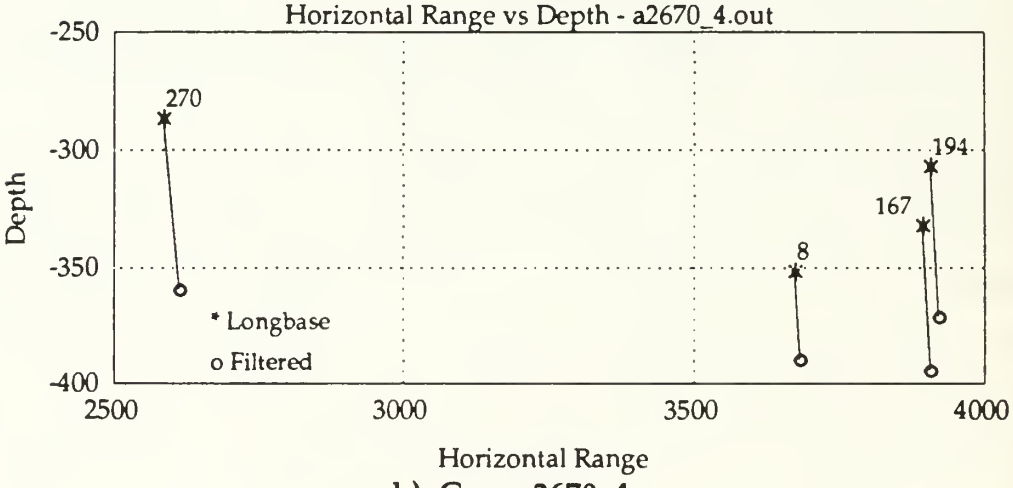
The graphs in Figure 8 accentuate the point made in Figure 7. The horizontal ranges are far too variable, ranging from under 2,000 feet to over 5,000 feet. In all cases there were three versions of track at common point counts, but not true "TOR data" in the anticipated sense.

These graphs illustrate some severe discrepancies. The magnitudes are quite variable, but the slopes are roughly of the same size. This seems to suggest that the errors may be dominated by the conditions in Figure 4 and perhaps, at a lower level, by the conditions in Figure 5.

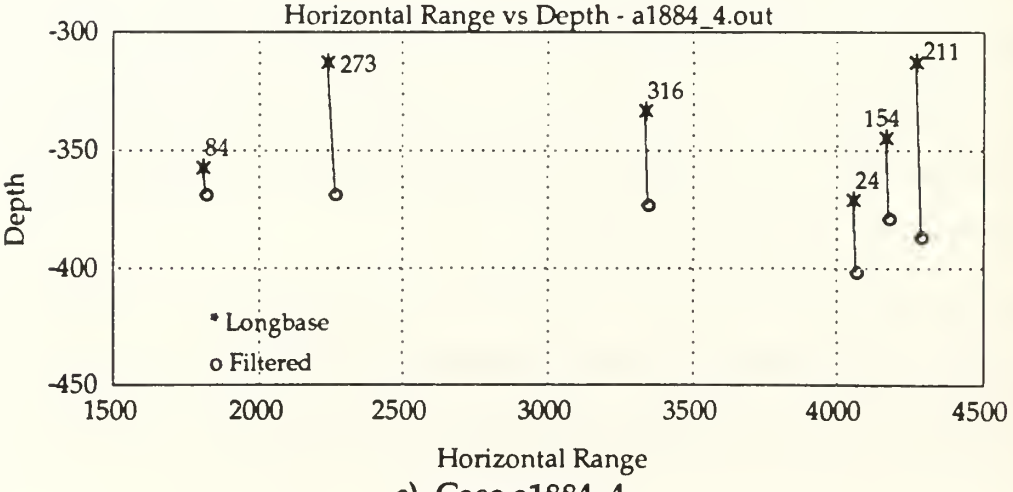
There could also be some tilt angle error activity because of the variability in the segment length. The case for this would be strengthened if the change in error sizes were a smooth function of azimuth after an adjustment for range. We



a) Case a2669_4.

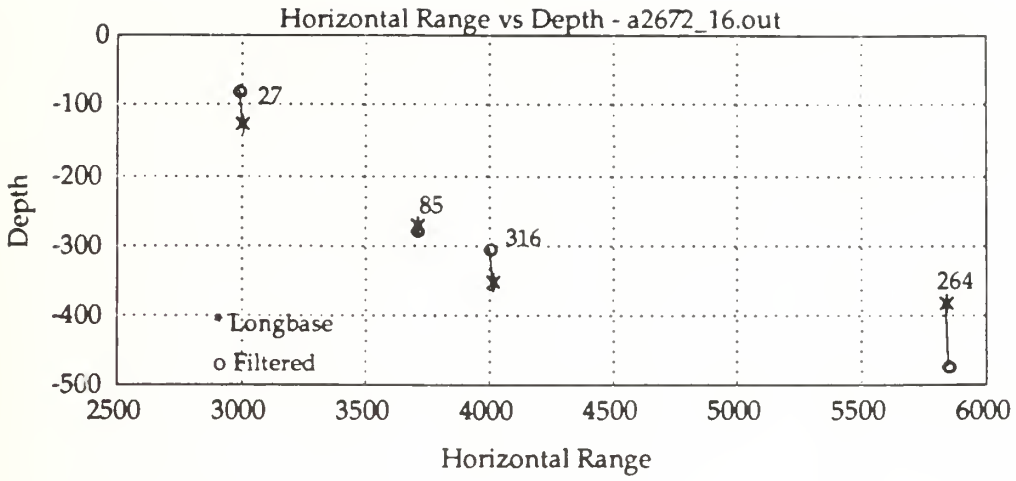


b) Case a2670_4

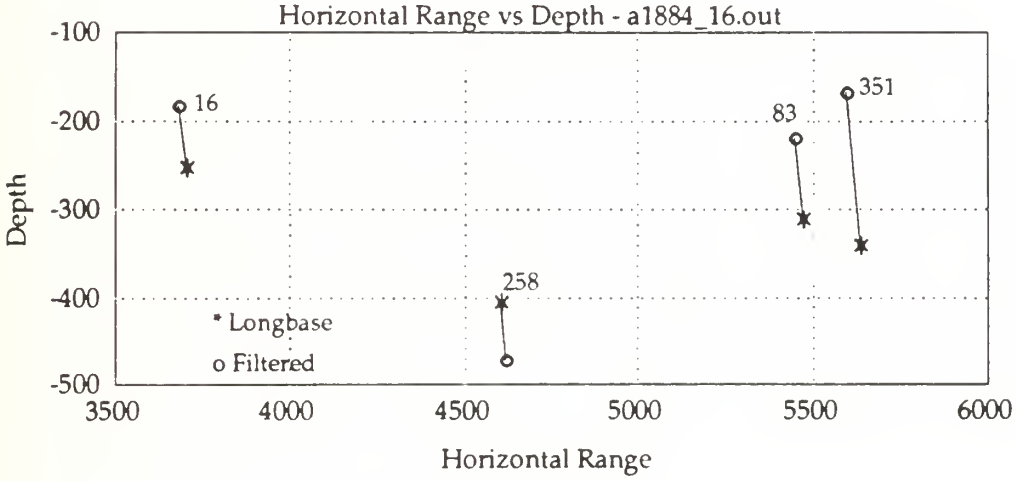


c) Case a1884_4

Figure 8. Horizontal Range vs Depth for Selected Cases



d) Case a2672_16



e) Case a1884_16

Figure 8 (continued). Horizontal Range vs Depth for Selected Cases

have yet to pursue further in this direction. However Table 1 does contain the slopes and azimuth angles for all cases in Figure 8.

Figure 9 contains situations that illustrate the use of fitting sine waves of the form

$$y = \mu + a\cos(\phi) + b\sin(\phi)$$

The details of fitting appear in Appendix A. Several error variables, horizontal range, elevation angles, etc., are periodic functions of azimuth measured from the base array. Sine wave fits allow estimation of the important characteristics of these periodic functions, specifically the offset μ, the amplitude and the phase angle. The coefficient of determination ($0 \leq CD \leq 1$) provides an indication of the quality of the fit, but the use of a single track does not leave many residual degrees of freedom to support it. Three degrees of freedom are expended in the estimation of μ, a, b.

The sine waves displayed are based upon 4 to 6 data points. These are too few for the drawing of reliable conclusions. We shall make a few comments however, just to illustrate the kinds of information that might be possible.

The horizontal range relative error for case 2669_4 has a strong *CD* of .9002; the phase angle appears to be near zero and μ = 8.2 ft per thousand feet suggests the presence of a horizontal offset. But we have two more cases with Array 4 as the base array. For the horizontal range relative error, these other two cases have phase angles of over 3 radians and μ values that are but half as large. The *CD* values are .9 and .46. These inconsistencies confirm that more data is needed. Similar inconsistencies show in other variables: relative vertical error, relative elevation angle error, and azimuth error.

The last two cases have Array 16 as the base array, and again, the inconsistencies can be charged to the sparseness of data. The sine wave plots do confirm the statements made surrounding the (h, z) plots, Figure 8, and have the potential of providing additional information about the elevation and azimuth angles. (The azimuth angles interact with the tilt angles and the horizontal displacement parameters. As yet there is no policy for dealing with this.)

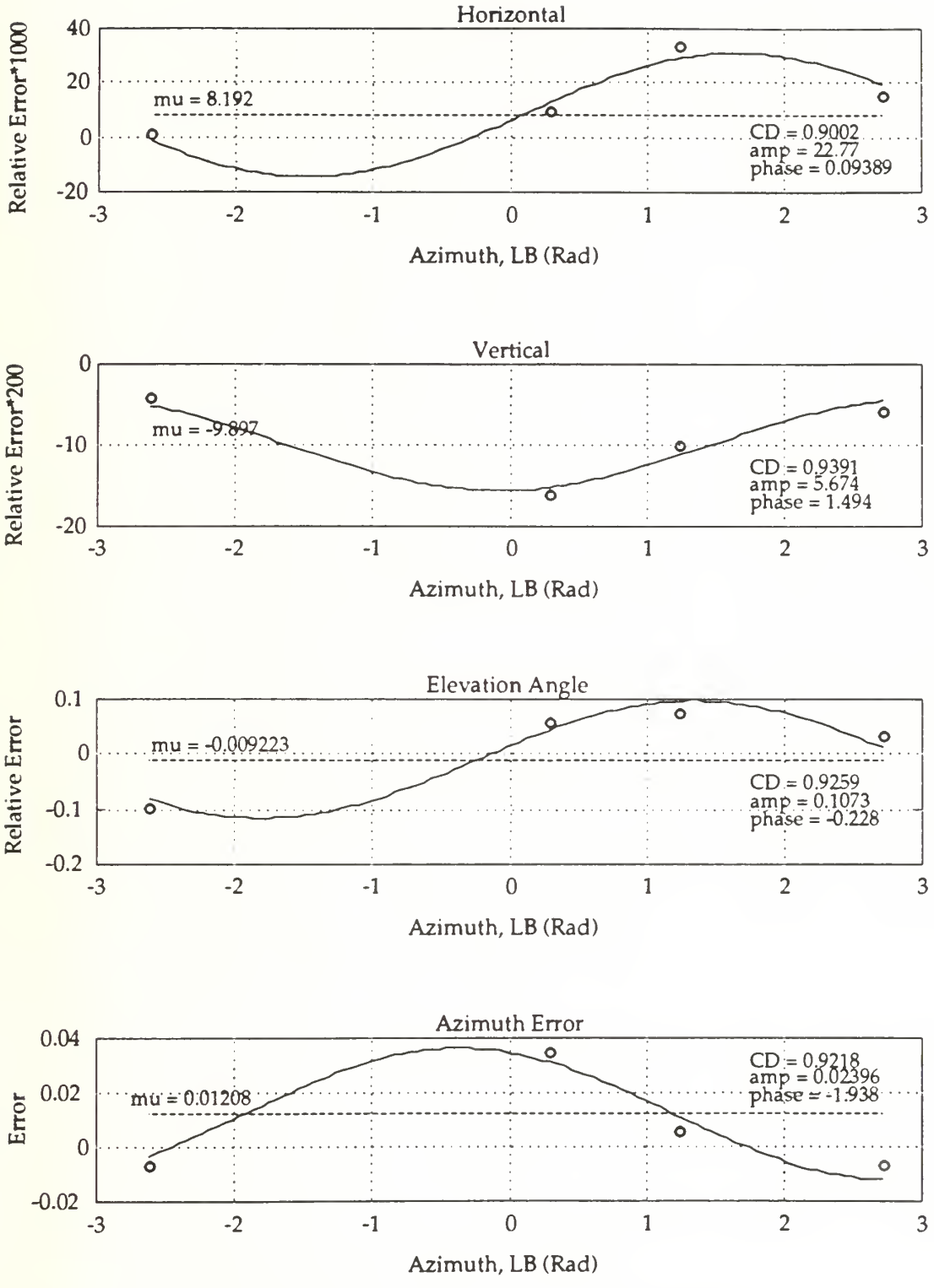


Figure 9a. Sine Wave Fits - Case a2669_4

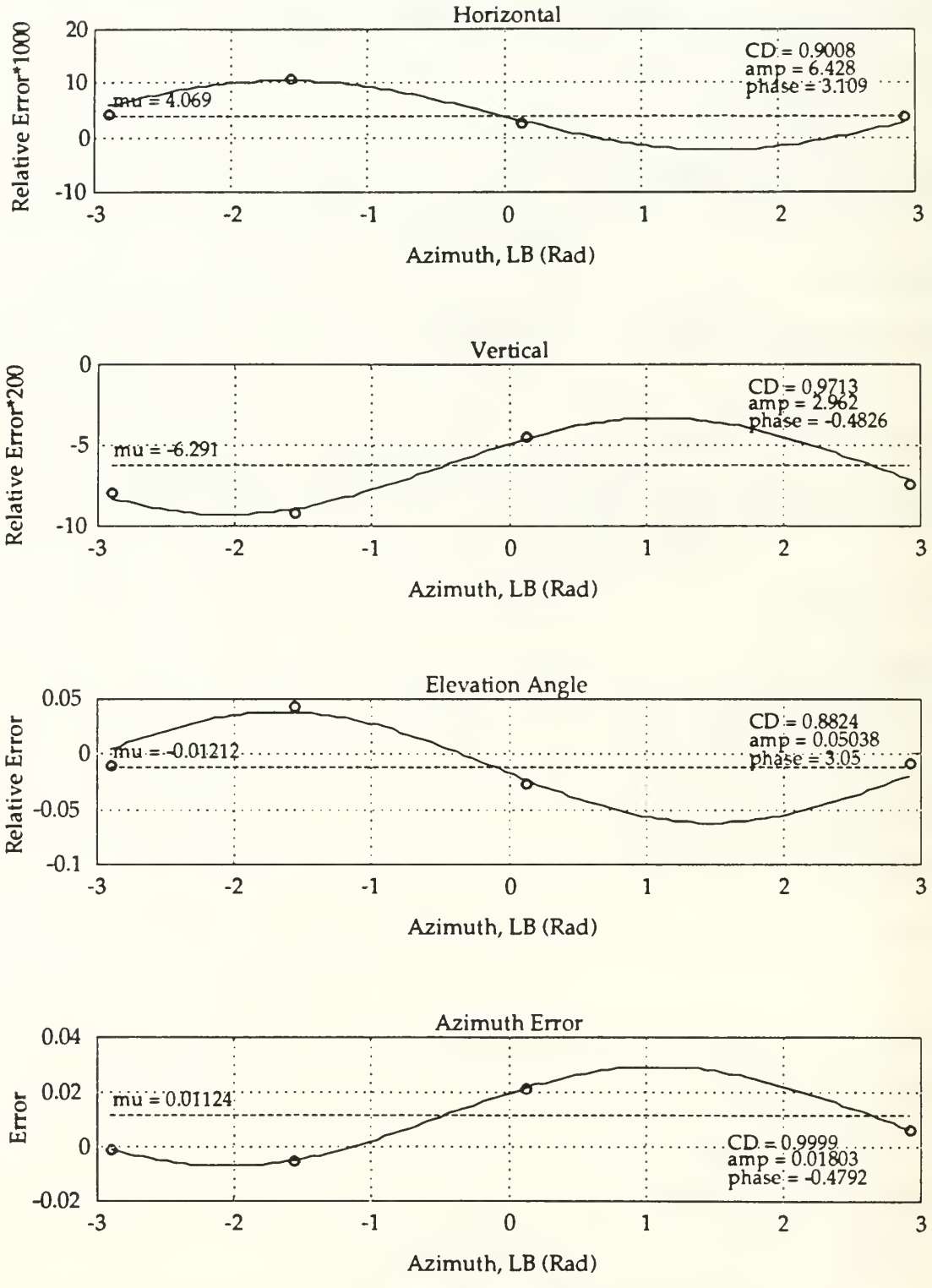


Figure 9b. Sine Wave Fits - Case a2670_4

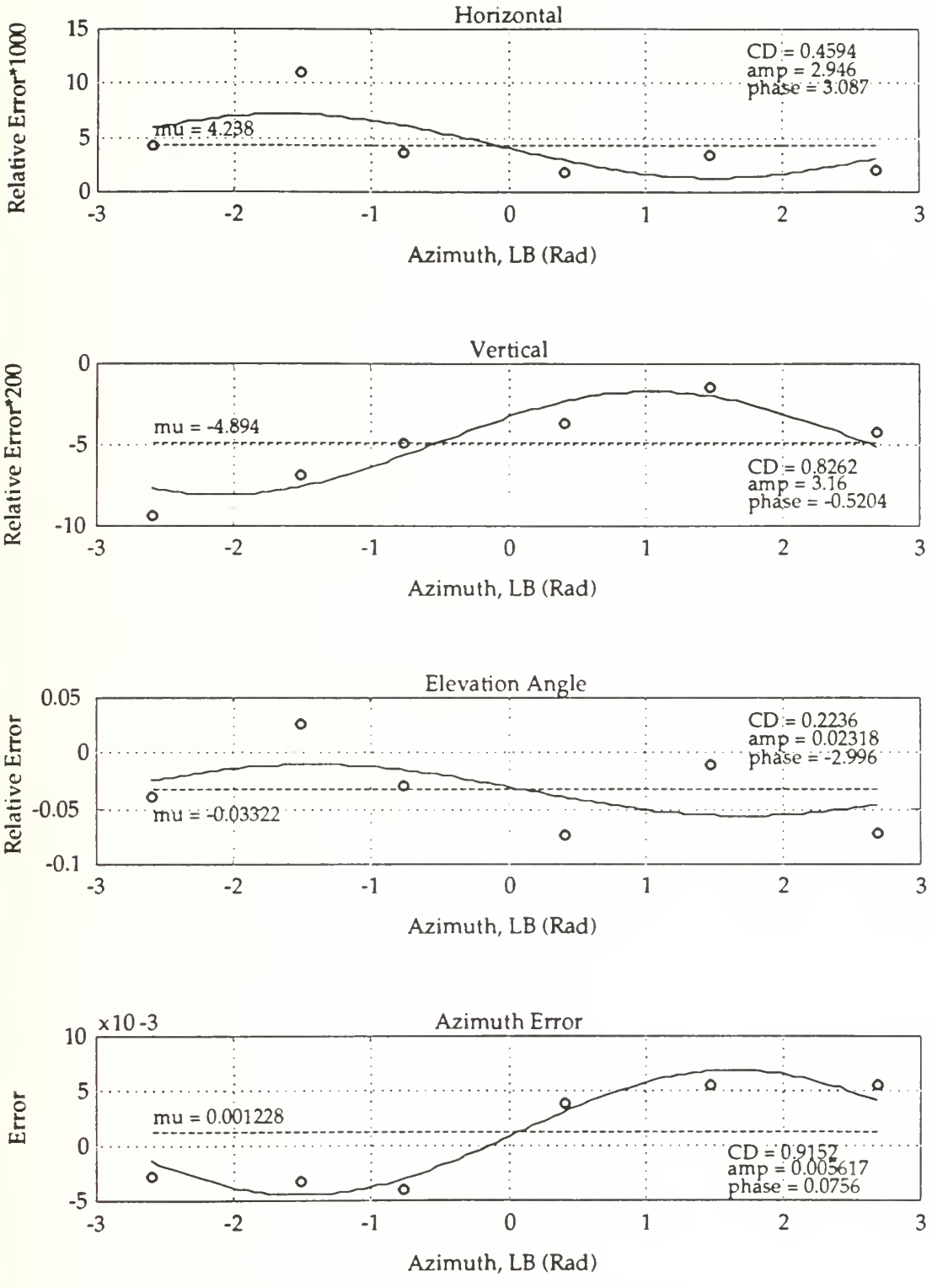
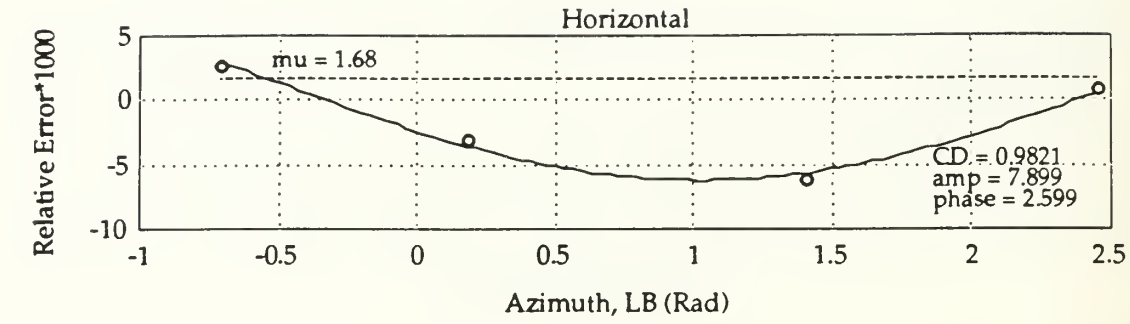
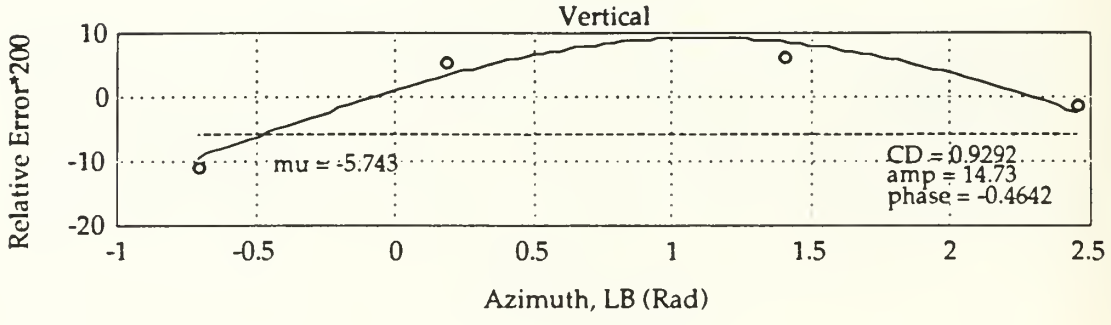
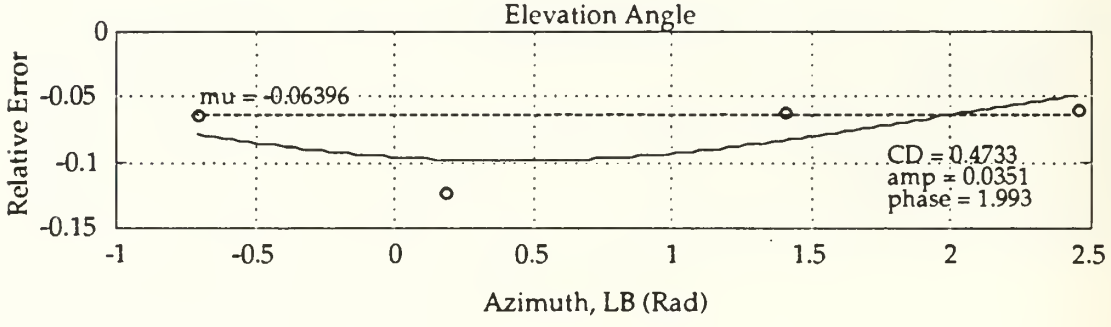


Figure 9c. Sine Wave Fits - Case a1884_4







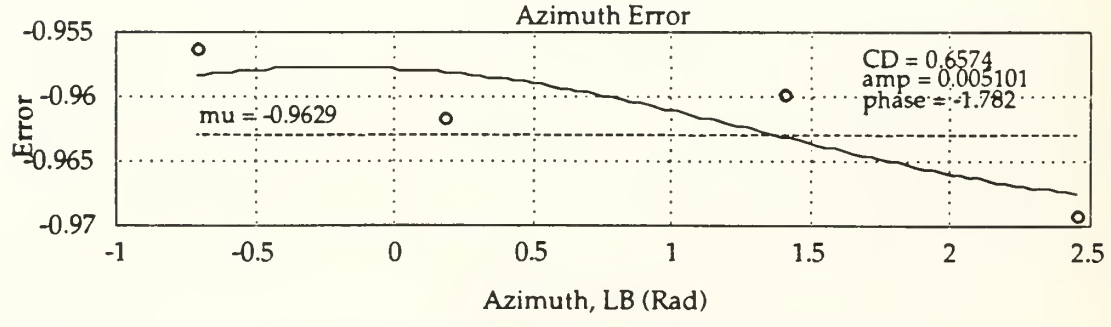


Figure 9d. Sine Wave Fits: Case a2672_16

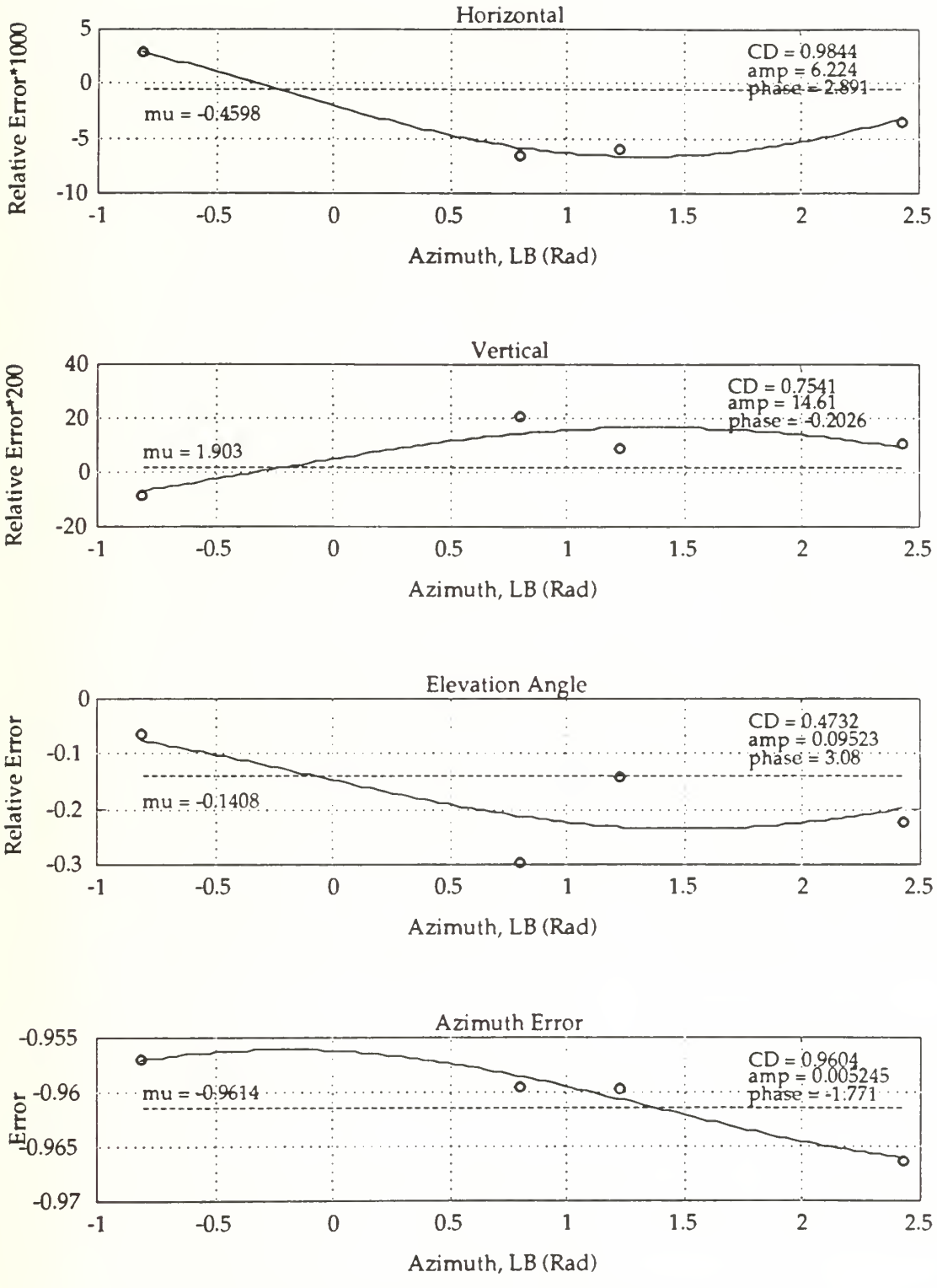


Figure 9e. Sine Wave Fits - Case a1884_16

We have yet to insert error information into the system of equations (1). The difficulty lies in finding usable values for the system of partial derivatives $\partial H/\partial z_c$, $\partial H/\partial \theta_i$, $\partial Z/\partial z_c$, $\partial Z/\partial \theta_i$. Some effort has been placed into a search for analytic approximations, but without success. It appears that we should go ahead with a numerical differentiation algorithm and leave the search for simplification to another time. This seems to be the most promising approach for treating the data at hand.

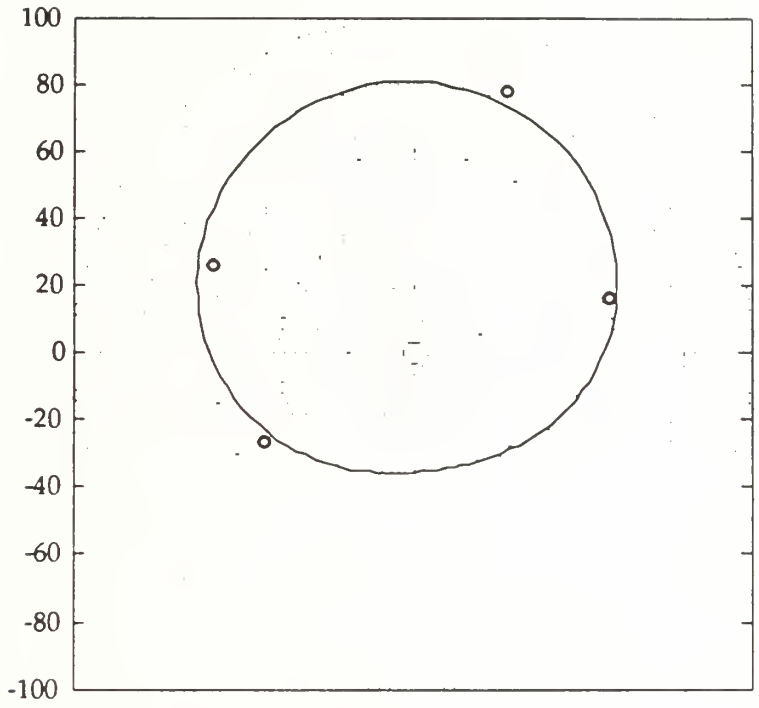
It is well known that a sine wave produces a circle when plotted in polar coordinates. Some exploratory work of this type was performed using our five cases but limited to the relative horizontal error times 1000 ft and the relative elevation angle error. These polar plots appear in Figure 10. Those plots dealing with relative horizontal error at 1000 ft do indeed resemble Figure 5. But the magnitude of the displacements cannot be determined from Figure 9 because, after all, they represent relative error and we do not know how to convert. Also, since the cases represent but two base arrays, we do not find consistency in the directions of the offsets based upon these plots. A like statement applies to the relative elevation angle plots. These plots provide an alternative to sine wave plots, but at this juncture, there seems to be no particular advantage.

Suggested Continuance

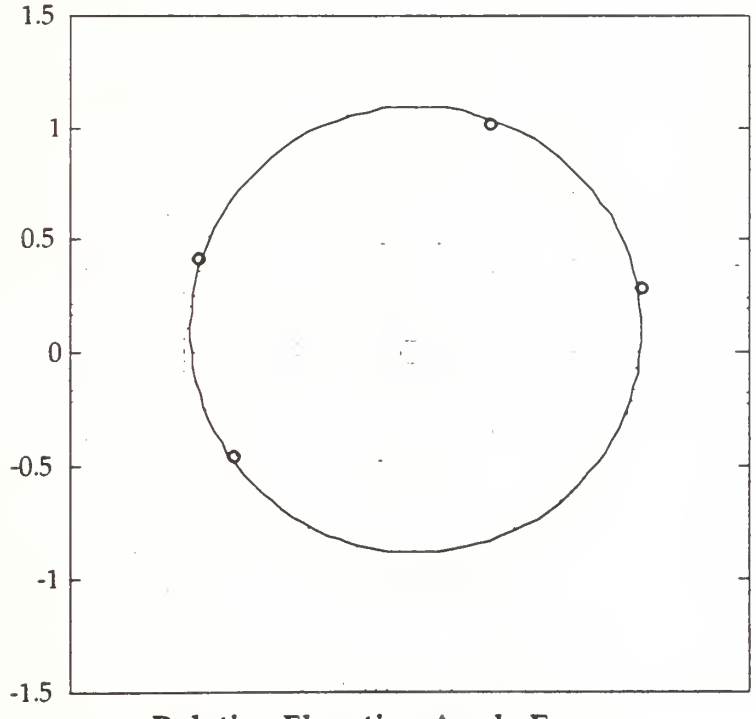
The following tactics seem appropriate based upon what we have learned thus far:

First. Much data is needed. The files should be scrubbed for as much pure TOR data as there is. If there is not enough, then data from DOR's might help out. The main issue here is to gather as much reference information as possible at 4,000 feet or more from the location of the base array. It will be necessary to use

Polar Plots: Relative Error

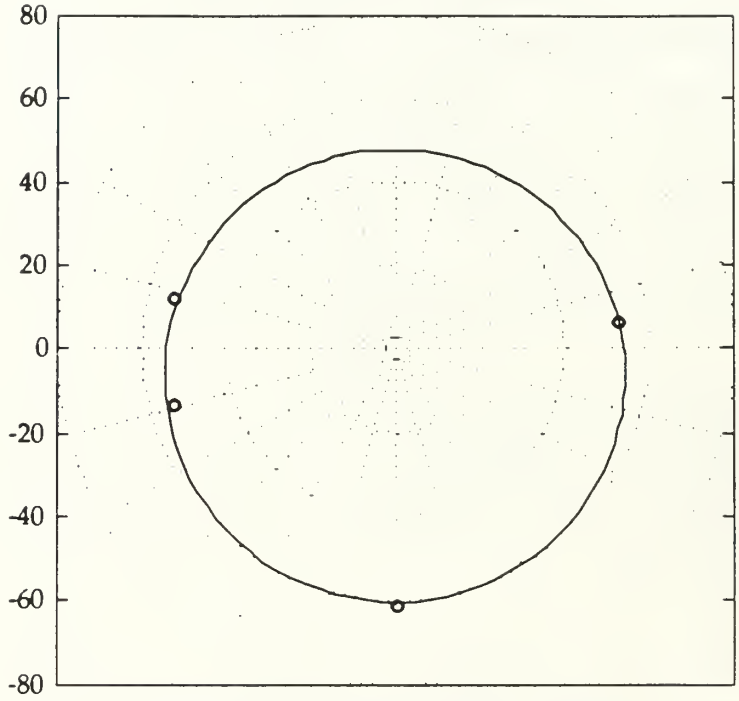


Relative Horizontal Error * 1000

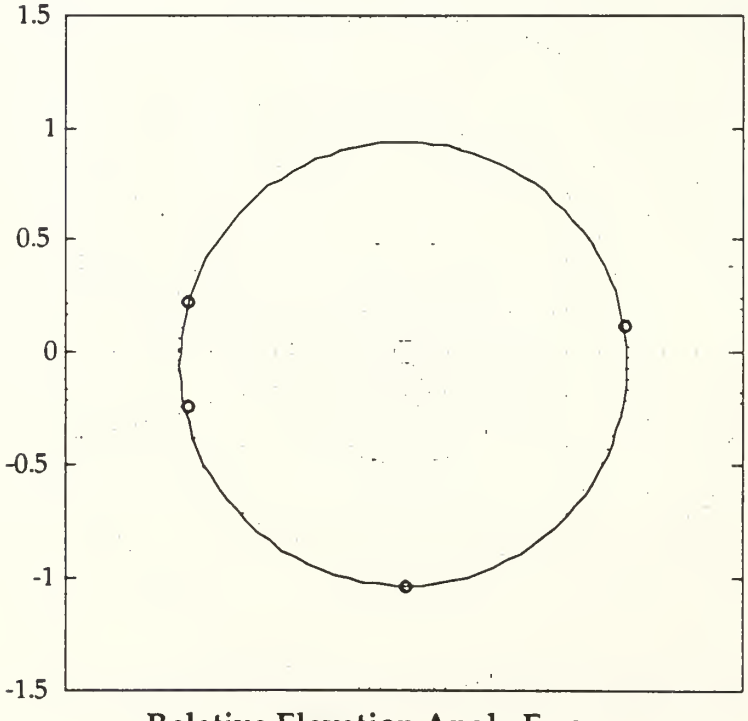


Relative Elevation Angle Error

Figure 10a. Case a2669_4

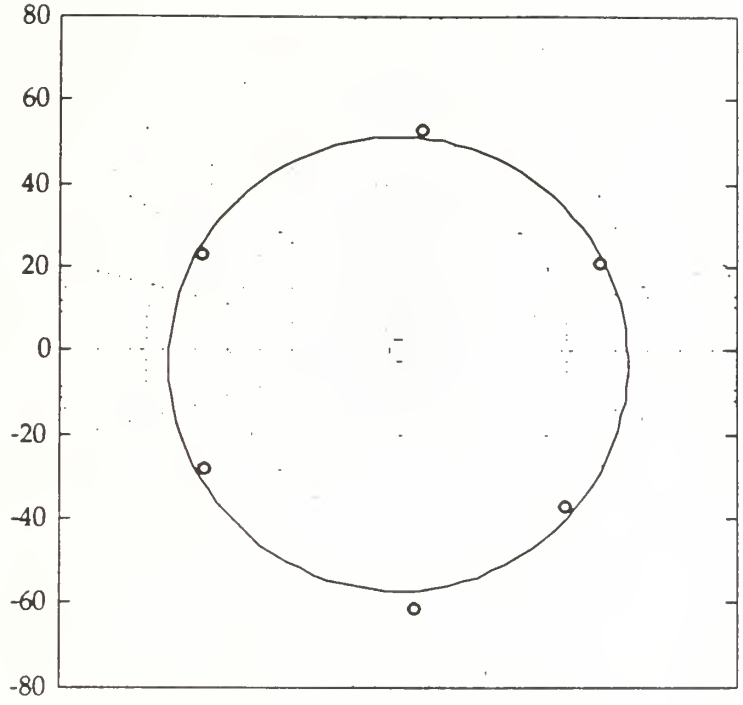


Relative Horizontal Error * 1000

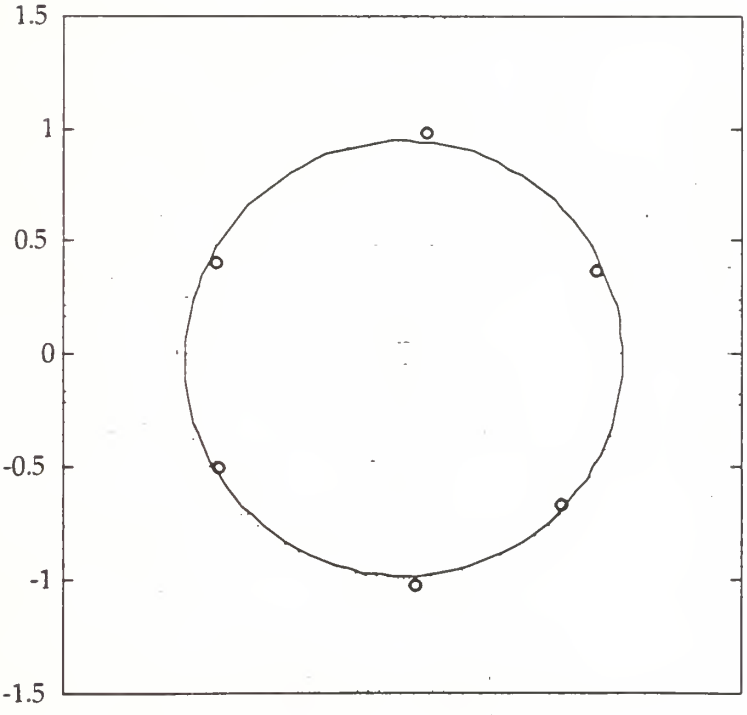


Relative Elevation Angle Error

Figure 10b. Case a2670_4

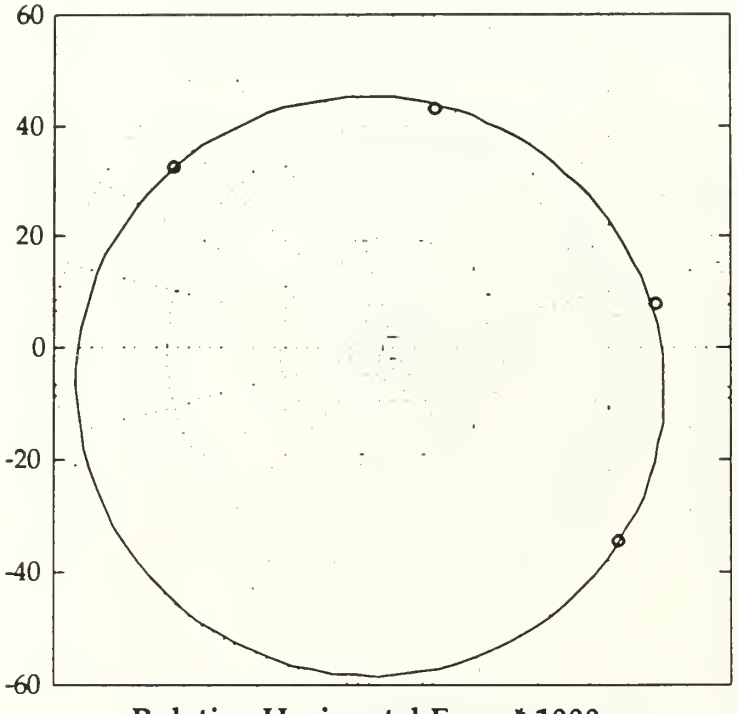


Relative Horizontal Error * 1000

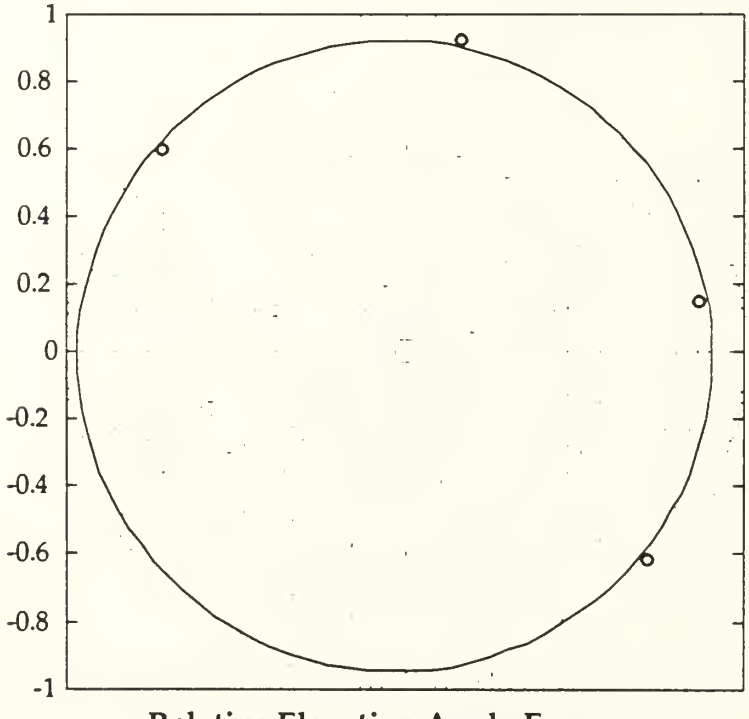


Relative Elevation Angle Error

Figure 10c. Case a1884_4

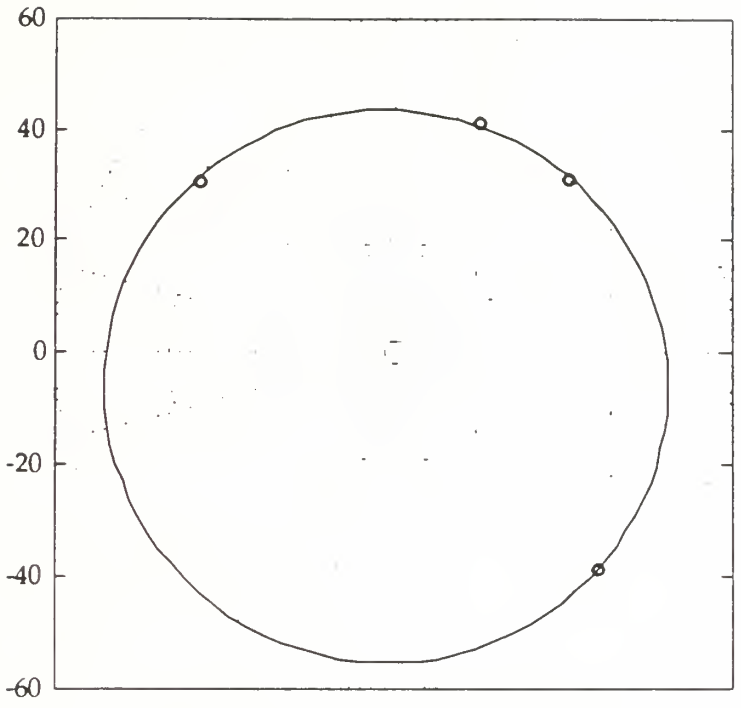


Relative Horizontal Error * 1000

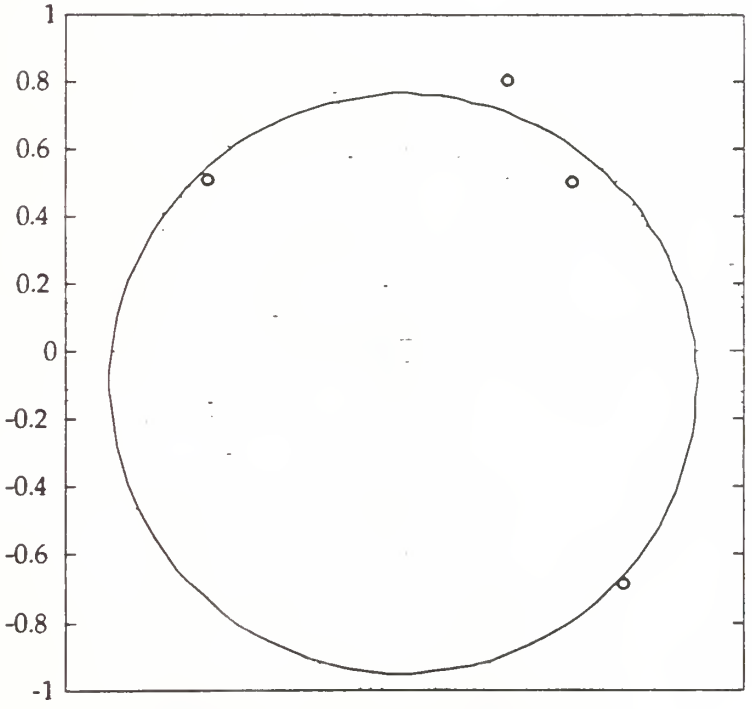


Relative Elevation Angle Error

Figure 10d. Case a2672_16



Relative Horizontal Error * 1000



Relative Elevation Angle Error

Figure 10e. Case a1884_16

different days in order to do this. It might be useful to group data according to the similarity of the water columns, at least to the extent possible. Data from the less variable water columns should be studied first.

Second. Suppose there is ample data for each TOR. Once the development of the diagnostic techniques has been completed, they should be applied to all and with every array taking its turn as the base array. Study of the results will begin in an interactive mode seeking the quality and location of the most promising corrective actions. Once we have identified a set of promising moves to make, it may be possible to automate some of the work.

Third. Having identified some corrective actions and estimated the quantitative changes, it may be wise to implement less than the full changes. The goal here is to discourage overswing. All of the diagnostics will be recomputed after the changes, and the interactive inspection process will begin again. Perhaps we could begin with 25% of the computed changes and measure the improvement of the separations.

Fourth. The role played by the LONGBASE algorithm should be low key at first and perhaps increase as we get closer to acceptable levels of track separations. This algorithm seems to be hyper sensitive to the array locations. Once the array locations are determined with but low levels of error, it should be possible to get excellent azimuth and elevation angles information using LONGBASE.

Fifth. Concurrent studies should be made concerning

(i) the *DV* column depth extrapolation method
(ii) the elevation angle that initializes the ray tracing algorithm
(iii) spatial interpolation of the *DV* column. Experiment with very simple one or two parameter models and convex interpolation methods and see if the track separations figures can be improved. This item would require that the ray tracing algorithm should be extended to include azimuth,

and hence become three dimensional. Up to now it has been independent of azimuth and two dimensional.

Data Extraction and Use

At this juncture we describe the data issues that were faced. A filter program was written for application to the NUWES raw time data files for extracting solely triple data points (three points with the same point count from different arrays). These triplets are associated with a TOR, identified by the contributing arrays, and at least 5 points must be present in a TOR for the data to be recorded. We were sent a tape containing the triple overlap raw timing data files as well as the corresponding position files (.TEX) and the sound velocity profile files (.CTD) for the dates of the runs.

Our first task was to inventory the data and present that inventory in ways convenient for selecting data sets by several different criteria. One criterion is the quantity and density of data in each TOR for each data file. We generate a histogram of the number of triplets found in each of the TORs (see Figure 11). To get an impression of the density of the points in each TOR we plot the point count vs. the array number (see Figure 12). From this 'time-line' plot we can see how dense and continuous the data are, what arrays are involved, the general direction of the vehicle, and the point count ranges of interesting data for extraction purposes.

It is necessary to identify those internal arrays that have good data in all six TORs, and over how many data sets we have good data. We use a map of the range for each data set and mark each TOR that contains data. At this point we have not considered the quality of the data in each TOR. The number of "available" TORs around each array is noted and transferred onto a spread sheet.

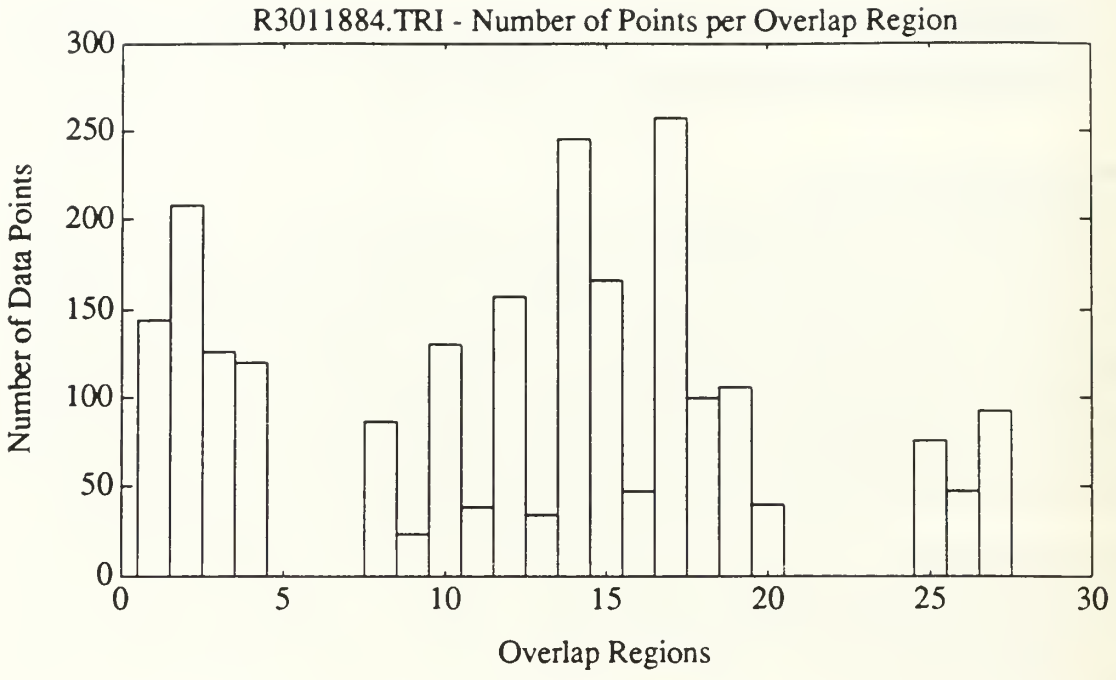


Figure 11

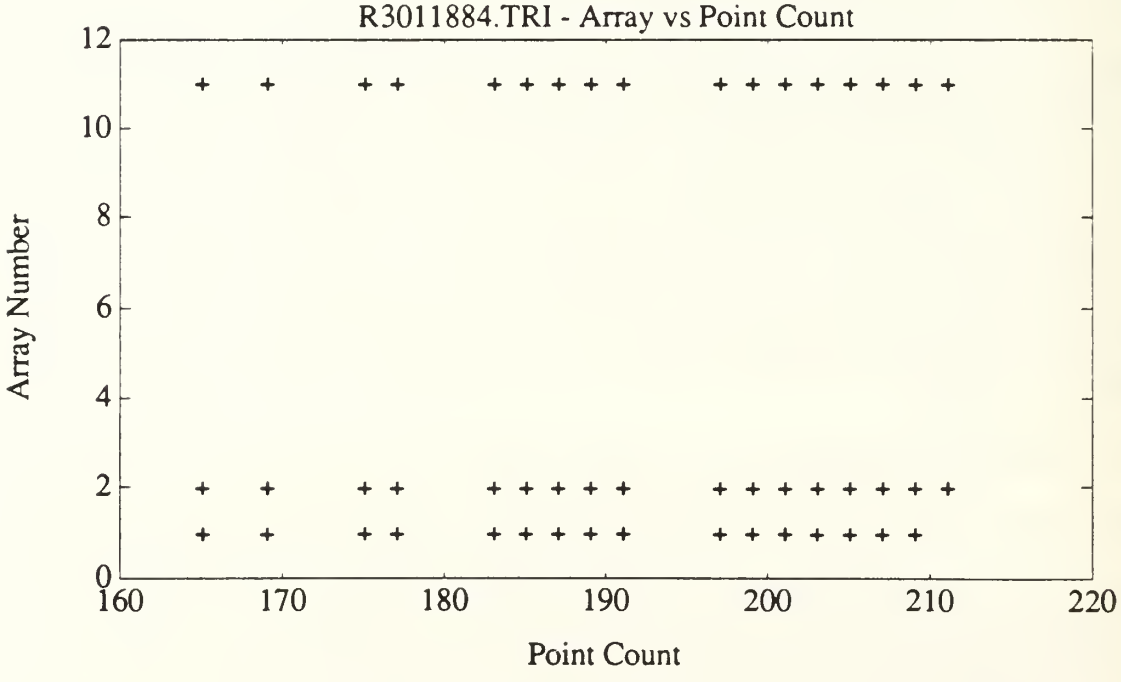


Figure 12

Including each of the data sets on the spread sheet allows us to generate a 3-D bar graph of the number of TORs around each array over all the data (see Figure 13). We can now easily select a base array with multiple satellite TORs for several days of data.

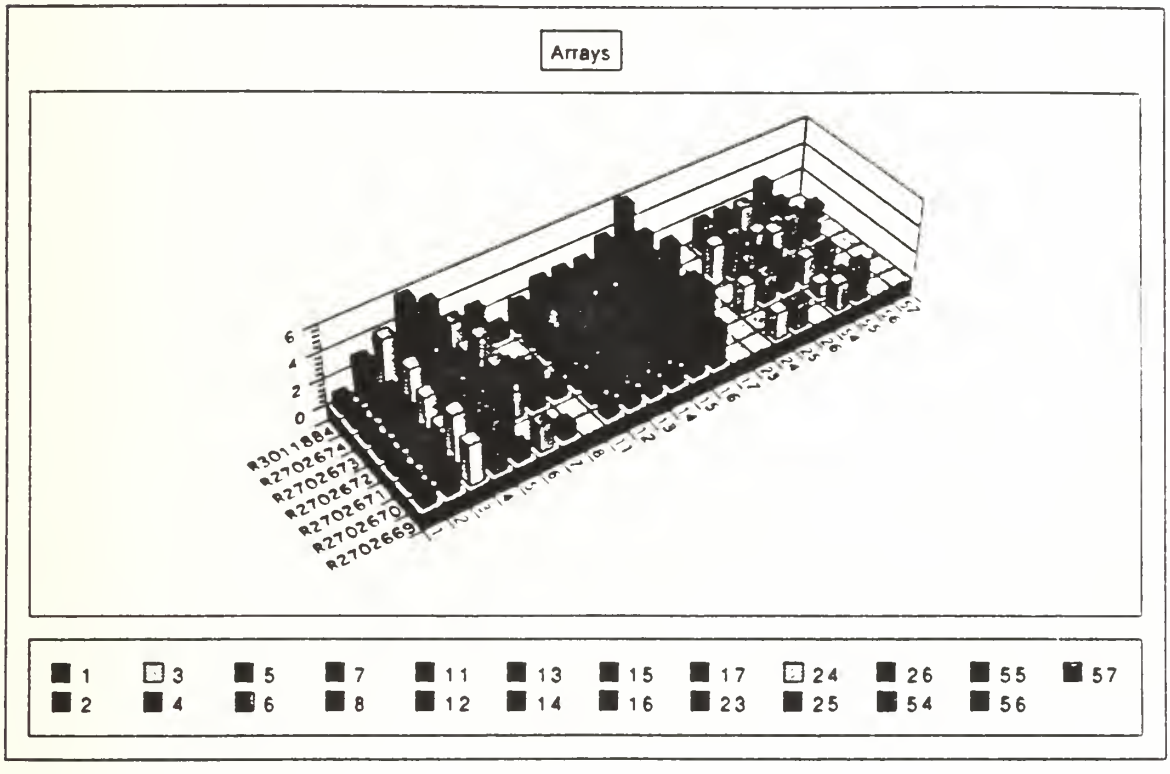


Figure 13

Currently the POSLOC/LONGBASE algorithm is using eleven data points for each of the three tracks in an overlap region. The choice of which eleven points out the possible hundreds in a TOR is fairly arbitrary as it is done by inspection. It is desirable to choose the points as nearly as possible to the center of the TOR, based on the intersection of arrival time curves. Such curves are shown in Figure 14, but this case serves to confirm the dilemma of Figure 7. Horizontal ranges cannot be roughly the same unless such is also true for transit times. The point counts within the selected range must be sequential.

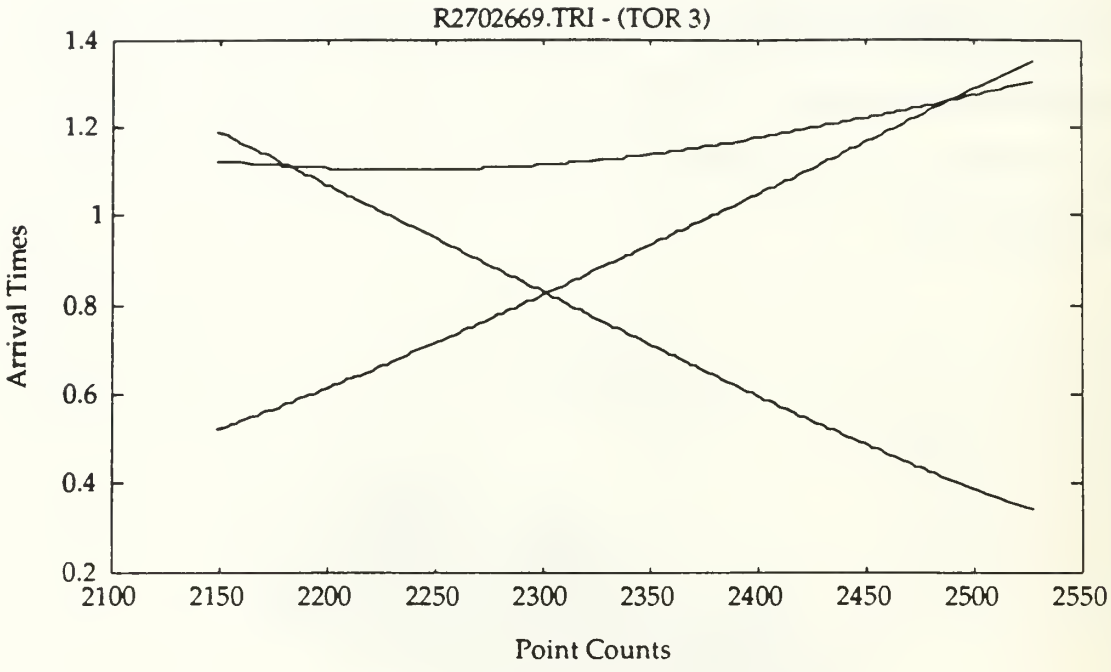


Figure 14

We have completed an "outlier rejection" algorithm to remove bad data points automatically, and we will be using more than just eleven data points for the POSLOC/LONGBASE algorithm. However, because of the constantly shifting needs for different testing, similar and dissimilar sets and overlaps in data, much of the data extraction and manipulation will still need to be done manually until we have more conclusive results on our algorithm testing.

REFERENCES

[1] Anderson, L, A., "Accurate Tracking for Short Baseline Hydrophone Arrays," NUWES, Keyport, WA 98345-0580, 1993.

[2] Beran, J., "Statistical Methods for Data with Long-Range Dependence," *Statistical Science*, **7**, November 1992.

[3] Camp, J. E. and Minto, R. W., "The St. Croix Three-Dimensional Underwater Tracking Range," APL-UW 6421, University of Washington, December 1968.

[4] Garrison, G. R. and Linger, E. H., "Effect of Bottom Currents on the 30-Foot Buoyant Tracking Array," APL-UW 6220, University of Washington, September 1962.

[5] Gembarski, J. A., "Use of Multiple Tracking Data in the Calibration of Short Baseline Arrays," NPS Master's Thesis, March 1992.

[6] Hurst, H. E., "Long Term Storage Capacity of Reservoirs," Transactions of the American Society of Civil Engineers, 77, 1951.

[7] Kirkland, P. C., Dewdney, C. J., and Francois, R. E., "Precision of FORACS Measurements," APL-UW 621, University of Washington, January 1968.

[8] Middleton, W. A., "Physical Oceanographic Characteristics of the Nanoose Range," NAVTORSTA Report 1163, NUWES, Keyport, WA, January 1973.

[9] Read, R. R., "A Technique for Assessing Short Baseline Array Tilt Errors," NPS Technical Report, NPS55-91-13, May 1991.

[10] Read, R. R., "A Study of Underwater Sound Ray Tracing Methodology," NPS Technical Report, NPS55-90-21, September 1990.

[11] Read, R. R., "An Investigation of Timing Synchronization Errors for Tracking Underwater Vehicles," NPS Technical Report, NPS55-90-15, July 1990.

[12] Read, R. R., "Program for the Simultaneous Estimation of Displacement and Orientation Corrections for Several Short Baseline Arrays," NPS Technical Report, NPS55-85-028, November 1985.

[13] Sandstrom, W. M., "Three-Dimensional Underwater Geometry and Survey Procedures," APL-UW 6917, University of Washington, August 1969.

APPENDIX A

Fitting a Sine Wave to Periodic Data

The data pairs are $\{x_i, y_i\}$ for $i = 1, \ldots, n$. The $\{x_i\}$ are in radians and, for our purposes, the period is 2π. We choose to fit a function of the form

$$y = \mu + a\cos(x) + b\sin(x)$$

using the principal of least squares. The goal is to choose μ, a, b so as to minimize the objective function.

$$\varphi = \sum_{i=1}^{n}\left[y_i - \mu - a\cos(x_i) - b\sin(x_i)\right]^2.$$

Taking the partial derivative φ_μ, φ_a, φ_b and setting them equal to zero leads to the Normal Equations:

$$\overline{y} = \mu + a\ \overline{\cos(x)} + b\ \overline{\sin(x)}$$

$$\Sigma y_i\cos(x_i) = \mu\,\Sigma\cos(x_i) + a\,\Sigma\cos^2(x_i) + b\,\Sigma\sin(x_i)\cos(x_i)$$

$$\Sigma y_i\sin(x_i) = \mu\,\Sigma\sin(x_i) + a\,\Sigma\sin(x_i)\cos(x_i) + b\,\Sigma\sin^2(x_i)$$

where the overbar notation refers to averaging, i.e. $\overline{g(x)} = \frac{1}{n}\sum_{i=1}^{n} g(x_i)$ for any function g.

The first normal equation can be substituted into the other two so as to eliminate μ from those two. Let us do this and use the notation

$$CC = \sum_{1}^{n}\left[\cos(x_i) - \overline{\cos(x)}\right]^2,$$

$$SC = \sum_{1}^{n}\left[\sin(x_i) - \overline{\sin(x)}\right]\left[\cos(x_i) - \overline{\cos(x)}\right],$$

$$SS = \sum_{1}^{n}\left[\sin(x_i) - \overline{\sin(x)}\right]^2.$$

Then these two equations can be expressed

$$\sum(y_i - \overline{y})\cos(x_i) = a\,CC + b\,SC$$

$$\sum(y_i - \overline{y})\sin(x_i) = a\,SC + b\,SS\ ,$$

a two by two linear system in a, b. A unique solution exists provided

$$\text{Det} = CC \cdot SS - (SC)^2 \neq 0$$

Let us call the solutions $\hat{a}, \hat{b}$ and use these values in the first normal equation to solve for

$$\hat{\mu} = \bar{y} - \hat{a}\ \overline{\cos(x)} + \hat{b}\ \overline{\sin(x)}$$

When $\hat{\mu}, \hat{a}, \hat{b}$ are substituted for μ, a, b in the original objective function φ, the result is called the sum of squared residuals:

$$\begin{aligned} SSR &= \sum_1^n \left[(y_i - \bar{y}) - \hat{a}\ \left(\cos(x_i) - \overline{\cos(x)}\right) - \hat{b}\ \left(\sin(x_i) - \overline{\sin(x)}\right) \right] \\ &= SSY - \sum_1^n \left[\hat{a}\ \left(\cos(x_i) - \overline{\cos(x)}\right) + \hat{b}\ \left(\sin(x_i) - \overline{\sin(x)}\right) \right]^2 \\ &= SSY - \left[\hat{a}^2\, CC + 2\,\hat{a}\, b^2\, SC + \hat{b}^2\, SS \right] \\ &= SSY - \left[\hat{a}\, Cy + \hat{b}\, Sy \right] \end{aligned}$$

where

$$\begin{aligned} SSY &= \sum_1^n (y_i - \bar{y})^2 \\ Cy &= \sum (y_i - \bar{y}) \cos(x_i) \\ Sy &= \sum (y_i - \bar{y}) \sin(x_i) \end{aligned}$$

and the normal equations play a prominent role in the algebraic establishment of the formulas above.

The quality of the fit is judged by CD, the coefficient of determination, which is the ratio of the variability with and without the model

$$CD = SSR / SSY = 1 - \frac{\hat{a}\, Cy + \hat{b}\, Sy}{SSY}$$

and $0 \leq CD \leq 1$.

Two of the more interesting outputs of the fit are the amplitude, A, of the sine wave; and the phase angle, ϕ_0, which locates the angle that the wave up crosses the level $\hat{\mu}$. The parameters are computed from

$$A = \sqrt{\hat{a}^2 + \hat{b}^2}$$

$$\phi_0 = ATAN2\left(\hat{a} / \hat{b}\right)$$

and the fitted wave has the form

$$\hat{y} = \hat{\mu} + A \sin\left(x - \phi_0\right).$$

SINFIT2.M is a function that returns the amplitude, phase, DC offset, and other parameters of the sine wave.

```
function [amp,phase,mu,CD,alpha,beta] = sinfit2(y,x)
% sinfit.m is a function to fit a sine wave to the data using
%       y = mu + alpha*cos(x) + beta*sin(x)
%
%  example:  [amp,phase,mu,CD] = sinfit(PHI_error,Phi_final)

% written by Colin R. Cooper          mod.: 12/21/92

y_ = mean(y);
c_ = mean(cos(x));
s_ = mean(sin(x));
cc = sum((cos(x)-c_).^2);
ss = sum((sin(x)-s_).^2);
sc = sum((sin(x)-s_).*(cos(x)-c_));
A = [cc sc;sc ss];
B = [sum((y-y_).*cos(x)); sum((y-y_).*sin(x))];
X = A\B;
alpha = X(1);
beta = X(2);
mu = y_ - alpha*c_ - beta*s_;
amp = sqrt(alpha^2 + beta^2);
phase = atan2(-alpha,beta);


SST = sum((y - y_).^2);
SSE = sum((y - alpha*cos(x) - beta*sin(x) - mu).^2);
CD = 1 - SSE/SST;
```

APPENDIX B

FILTER3.FOR is a pre-processing filter applied to the raw arrival times data files. This program was used at NUWES on the raw data files to generate output data files which were sent to NPS for further processing.

FILTER3.FOR reads the input and output data filenames from the ASCII file, FNAME.LST, which is prepared by the user is accordance with the operating system naming conventions. The file header is transferred directly to the output file, but the sound velocity, array, and hydrophone records are not transferred. The point count and PSK records are read and, if three sequential records have the same point count, they are written to a temporary storage file.

Once all the possible triplets have been written to the temporary file, it is read from the beginning, recording the occurrence of each triplet according to it's TOR, beginning and ending point counts, and the number of occurrences.

FILTER3.FOR makes another pass on the temporary data file checking the data points against the triplet TOR and point count information. When there are five (5) or more occurrences in a TOR the data for that region is written to the output data file.

FORTRAN SOURCE CODE

```
      PROGRAM FILTER3
C*************************************************************
C  FILTER3.FOR  written by Colin R. Cooper          6/1/92
C ------------------------------------------
C   - READS THE INPUT FILE NAMES FROM A DIRECTORY FILE CALLED
C       'FNAME.LST'
C
C STAGE 1:
C    PROGRAM FOR READING OFF THE TIMES FROM THE RAW DATA FILE.
C  WE WILL ONLY BE READING THE TIMES ASSOCIATED WITH MODES 7+8.
C       - TAKES THE HEADER RECORD AND STORES IT TO THE OUTPUT
C         DATA FILE READ FROM FILENAME.LST.
C       - SKIPS THE SOUND VELOCITY, ARRAY, AND HYDROPHONE
C         RECORDS.
```

```
C       - READS THE POINT COUNT RECORDS.
C       - READS THE PSK DATA RECORDS.
C       - STORES THE TRIPLE OVERLAP DATA IN THE FOLLOWING
C         FORMAT TO A TEMPORARY FILE.
C
C      FIELD              DATA                              DATA
C     POSITION            NAME              SPECIF.         DESCRIPTION
C-------------------------------------------------
C        1-5           POINT_COUNT            I5          3D Processed Point Count
C        6-9             ARRAY                I4          Array Code
C       10-13           OBJECT                I4          Vehicle ID
C       14-23           X_TIME                I10         Arrival Time to X-
C                                                            Hydrophone
C       24-33           Y_TIME                I10         Arrival Time to Y-
C                                                            Hydrophone
C       34-43           Z_TIME                I10         Arrival Time to Z-
C                                                            Hydrophone
C       44-53           C_TIME                I10         Arrival Time to C-
C                                                            Hydrophone
C
C
C STAGE 2:
C   READS THE TRIPLE OVERLAP RECORDS FROM THE TEMP FILE AND
C   FILTERS OUT ALL TRIPLE OVERLAP POINTS THAT DON'T MEET THE
C   FOLLOWING CRITERIA:
C      1) AT LEAST 5 POINTS IN THE SAME OVERLAP REGION.
C
C STAGE 3:
C   READS THE TEMP1.DAT FILE, CHECKS THE STATS ON THE OVERLAP
C   REGION AND WRITES THE RECORD TO OUTPUT FILE IF THE RULES
C   ARE MET.
C*************************************************************

      DIMENSION LINE(55),LINEPC(5),LINPS7(3,55),LINPS8(3,55)
      INTEGER T1(3),T2(3),T3(3),T4(3),PC(3),ARR(3),MODE
      INTEGER CT7,CT8,OVREG(27),OVCOUNT(27,2),PCCNT(27,2,2)
      INTEGER OVNUM
      CHARACTER FNAME*30,FIN*25,FOUT*25

      DATA OVREG/648,11880,1100,15400,1680,19152,2394,22,264,72,
     *       468,156,728,280,1050,450,1440,672,1904,952,4186,
     *       7728,5040,9000,6000,10400,7072/

C---------
C STAGE 1:
C---------

C OPEN ALL NECESSARY FILES
  5  OPEN(UNIT=2,FILE='FILENAME.LST',STATUS='OLD')
  7  READ(2,1020,ERR=980,END=990)FIN,FOUT
     OPEN(UNIT=3,FILE=FIN,STATUS='OLD')
     OPEN(UNIT=4,FILE='TEMP1.DAT',STATUS='NEW')
```

```
      OPEN(UNIT=5,FILE=FOUT,STATUS='NEW')

C   READ THE SECURITY RECORD AND THE HEADER RECORD AND
C     WRITE THE HEADER RECORD TO THE OUTPUT FILE.
      READ(3,1000,ERR=70,END=80)(LINE(I), I = 1,55)
      READ(3,1000,ERR=70,END=80)(LINE(I), I = 1,55)
      WRITE(5,1000)(LINE(I), I = 1,55)

C   START LOOP FOR FILTERING OUT THE DATA INTO TRIPLE OVERLAP
  10  READ(3,1000,ERR=70,END=80)(LINE(I), I = 1,55)
      IF(LINE(2).EQ.'C') THEN
        DO 20 I = 1,5
  20      LINEPC(I)=LINE(I+3)
        CT7 = 1
        CT8 = 1
      ELSEIF((LINE(2).EQ.'S').AND.(LINE(10).EQ.'7')) THEN
        DO 30 I = 1,55
  30      LINPS7(CT7,I)=LINE(I)
        CT7 = CT7 + 1
      ELSEIF((LINE(2).EQ.'S').AND.(LINE(10).EQ.'8')) THEN
        DO 40 I = 1,55
  40      LINPS8(CT8,I)=LINE(I)
        CT8 = CT8 + 1
      ENDIF
      IF(CT7.EQ.4) THEN
        DO 50 I = 1,3
  50      WRITE(4,1010)(LINEPC(J), J = 1,5),(LINPS7(I,K),K = 3,50)
        CT7 = 1
      ELSEIF(CT8.EQ.4) THEN
        DO 60 I = 1,3
  60      WRITE(4,1010)(LINEPC(J), J = 1,5),(LINPS8(I,K),K = 3,50)
        CT8 = 1
      ENDIF
      GOTO 10

C IF AN ERROR OCCURED IN READING THE DATA FILE, RECORD THE ERROR
C   AND CONTINUE ON WITH THE NEXT DATA FILE.
  70  WRITE(*,*)'   THERE WAS AN ERROR READING THE DATA.'
      WRITE(*,*)'   CONTINUING WITH THE NEXT DATA FILE !'
      WRITE(5,*)'   ERROR READING DATA IN FILE',FIN
      CLOSE(UNIT=3)
      CLOSE(UNIT=4,STATUS='DELETE')
      CLOSE(UNIT=5)
      GOTO 7

C STATE 1 WAS COMPLETED NORMALLY, CLOSE THE DATA FILE AND REWIND
C   THE TEMPORARY STORAGE FILE.
  80  CONTINUE
      CLOSE(UNIT=3)
      REWIND(4)
```

```
C-------------------------------------------
C STAGE 2: READ THE RECORD AND STORE THE NUMBER OF
C         OCCURENCES OF THE OVERLAP REGION IN AN ARRAY.
C-------------------------------------------

C SET THE STATISTICS ARRAYS TO ZEROS
 160  DO 190 I = 1,27
       DO 180 J = 1,2
         DO 170 K = 1,2
 170       PCCNT(I,J,K) = 0
 180     OVCOUNT(I,J) = 0
 190  CONTINUE

C READ IN A TRIPLE OVERLAP SET AND INCREMENT THE APPROPRIATE
C   COUNTER FOR THE REGION AND MODE.
 200  DO 210 I=1,3
 210  READ(4,1030,ERR=230,END=300)PC(I),ARR(I),MODE,T1(I),T2(I),
     *     T3(I),T4(I)
      OVNUM = ARR(1)*ARR(2)*ARR(3)
      I = 1
 220  IF (OVNUM.EQ.OVREG(I)) THEN
        OVCOUNT(I,MODE-6) = OVCOUNT(I,MODE-6) + 1
        IF (OVCOUNT(I,MODE-6).EQ.1) THEN
          PCCNT(I,MODE-6,1)=PC(1)
        ELSE
          PCCNT(I,MODE-6,2)=PC(1)
        ENDIF
        GOTO 200
      ENDIF
      I = I+1
      IF(I.GT.27) GOTO 200
      GOTO 220

C IF AN ERROR IS FOUND IN THE TEMP FILE, RECORD IN THE OUTPUT FILE
C   AND GO TO THE NEXT DATA FILE.
 230  WRITE(5,*)'ERROR READING TEMP1.DAT FILE'
      CLOSE(UNIT=4,STATUS='DELETE')
      CLOSE(UNIT=5)
      GOTO 7

 300  CONTINUE
      REWIND(4)

C-------------------------------------------
C STAGE 3: READ THE RECORD AND DECIDE WHICH DATA POINTS
C         TO KEEP AND WRITE TO THE OUTPUT FILE.
C-------------------------------------------

C WRITE STATISTICS AT THE TOP OF THE FILE.
      WRITE(5,1040)
      DO 350 I=1,27
        DO 340 J = 1,2
```

```
        IF(OVCOUNT(I,J).GT.0) THEN
          WRITE(5,1050)I,J+6,OVCOUNT(I,J),PCCNT(I,J,1),
     * PCCNT(I,J,2)
        ENDIF
  340  CONTINUE
  350  CONTINUE
       WRITE(5,*)' '

C START WRITING DATA RECORDS IN THE ABOVE FORMAT IF ENOUGH DATA
C   POINTS ARE AVAILABLE.
  400  DO 410 I=1,3
  410  READ(4,1030,ERR=490,END=500)PC(I),ARR(I),MODE,T1(I),T2(I),
     * T3(I),T4(I)
       OVNUM = ARR(1)*ARR(2)*ARR(3)
       I = 1
  420  IF (OVNUM.EQ.OVREG(I)) THEN
         IF (OVCOUNT(I,MODE-6) .GE. 5) THEN
           DO 430 J=1,3
  430        WRITE(5,1030)PC(J),ARR(J),MODE,T1(J),T2(J),
     *  T3(J),T4(J)
         ENDIF
         GOTO 400
       ENDIF
       I = I+1
       IF(I.GT.27) GOTO 400
       GOTO 420

  490  WRITE(5,*)'THERE WAS AN ERROR READING THE TEMP1.DAT FILE'
  500  CLOSE(UNIT=4,STATUS='DELETE')
       CLOSE(UNIT=5)
       GOTO 7

C-------------------------------
C RETURN TO THE TOP AND START ON THE NEXT DATA FILE.
C-------------------------------

  980  WRITE(5,*)' THERE WAS AN ERROR READING THE DIR.LST FILE'
  990  CONTINUE
       CLOSE(UNIT=2)

 1000 FORMAT(55A1)
 1010 FORMAT(5A1,48A1)
 1020 FORMAT(A12,1X,A12)
 1030 FORMAT(I5,2I4,4I10)
 1040 FORMAT(4X,'OV REG',4X,'MODE',4X,'NUM PTS',4X,'FIRST PC',3X,
     * 'LAST PC')
 1050 FORMAT(6X,I2,7X,I2,3(6X,I5))
       STOP
       END
```

APPENDIX C

The following are MATLAB programs used for processing and presenting input and output data from the algorithms being tested. These programs must be run from within the MATLAB environment.

LINPLT2.M generates Figure 8.

```
% LINPLT2.M  plots the H1 vs Depth and Hlongbase vs Depth with
%           all six (max) overlap regions plotted together.
%
%         [slope,ang] = linplt1('filename',base_aray)
%function [slope,ang] = linplt1(fname1,AC)
clear
 fname1=input('Enter the data File Name » ','s');
AC = input('Enter the base array Num >> ');
eval(['load ',fname1]);
fname2 = fname1(1:length(fname1)-4);
%eval(['!del ',fname2,'.met;']);
load array
X1 = eval([fname2,'(:,4)']);
Y1 = eval([fname2,'(:,5)']);
Xc = xpos(find(num==AC));
Yc = ypos(find(num==AC));

H1 = eval([fname2,'(:,18)']);
Hlb = eval([fname2,'(:,19)']);
Z1 = eval([fname2,'(:,6)']);
Zlb = eval([fname2,'(:,15)']);
Theta = eval([fname2,'(:,20)']);
Thetalb = eval([fname2,'(:,21)']);
clg
plot(H1,-Z1,'ow',Hlb,-Zlb,'*w',[H1';Hlb'],[-Z1';-Zlb'],'-w'),grid
title(['Horizontal Range vs Depth - ',fname1])
xlabel('Horizontal Range'),ylabel('Depth')
text(.2,.2,'* Longbase','sc')
text(.2,.17,'o Filtered','sc')
for i = 1:length(H1)
  ang(i) = atan2(Y1(i)-Yc, X1(i)-Xc);
  slope(i) = (-Z1(i)+Zlb(i))/(H1(i)-Hlb(i));

  s1 = sprintf('%1.0f',rad2deg(ang(i)));
  text(Hlb(i),-Zlb(i),s1)

end
disp(rad2deg(ang))
disp(H1')
disp( (H1 - Hlb)')
```

```
disp( Z1')
disp((Z1 - Zlb)')
disp(slope)
disp(Theta')
disp((Theta - Thetalb)')
disp(((H1 - Hlb)./Hlb)')
disp(((Z1 - Zlb)./Zlb)')
disp(((Theta - Thetalb)./Thetalb)')
```

LGBSPLT4.M generates the sine fitted plots of Figure 9.

```
% LGBSPLT4.M  plots the Azimuth, Horizontal ranges, and Elevation
%            Angles plots for each overlap region.
%    written by Colin R. Cooper            mod.: 5/3/93

clear
fname1=input('Enter the data File Name » ','s');
eval(['load ',fname1]);
fname2 = fname1(1:length(fname1)-4);
%eval(['!del ',fname2,'.met;']);

arrays=eval([fname2,'(:,1:3)']);
P1 = eval([fname2,'(:,4:6)']);
P2 = eval([fname2,'(:,7:9)']);
P3 = eval([fname2,'(:,10:12)']);
LBP = eval([fname2,'(:,13:15)']);
data = eval([fname2,'(:,16:33)']);
Pzs = [P1(:,3) P2(:,3) P3(:,3)];
C = input('Enter Center Array number » ');

Zfilt = [];
PHI = [];
H = [];
THETA = [];
ind = find(arrays == C);
for i = 1:length(ind)
  n(i) = find(arrays(i,:)==C);
  PHI = [PHI ; data(i,n(i)*6-5:n(i)*6-4)];
  H = [H ; data(i,n(i)*6-3:n(i)*6-2)];
  THETA = [THETA ; data(i,n(i)*6-1:n(i)*6)];
  Zfilt = [Zfilt ; Pzs(i,n(i))];
  Pavg(i,:) = mean([P1(i,:); P2(i,:); P3(i,:)]);
end

Pbar = [((sum(((P1-Pavg).^2)')').^ .5) ((sum(((P2-Pavg).^2)')').^ .5) ...
      ((sum(((P3-Pavg).^2)')').^ .5)];
D = sum(Pbar')';
PbarLBP = [((sum(((P1-LBP).^2)')').^ .5) ((sum(((P2-LBP).^2)')').^ .5) ...
      ((sum(((P3-LBP).^2)')').^ .5)];
LBD = sum(PbarLBP')';
```

```
% Prepare the plots:
load array
Zc = zpos(find(num == C));
clg

%  Horizontal Relative Error*1000:
H_rel = 1000*(H(:,1)-H(:,2))./H(:,2);
[amp,phase,mu,CD,alpha,beta] = sinfit2(H_rel,PHI(:,2));
%[amp phase mu CD alpha beta]
PHImin = min(PHI(:,2));
PHImax = max(PHI(:,2));
w = linspace(PHImin,PHImax,100);
s = amp*sin(w - phase) + mu;
subplot(211),plot(PHI(:,2),H_rel,'ow',w,s,'-w',...
        [PHImin PHImax],[mu mu],'-w'),grid
text(PHImin,mu,['mu = ',num2str(mu)]);
text(.75,.9,['CD = ',num2str(CD)],'sc');
text(.75,.87,['amp = ',num2str(amp)],'sc');
text(.75,.84,['phase = ',num2str(phase)],'sc');
title('Horizontal'),xlabel('Azimuth, LB (Rad)'),ylabel('Relative Error*1000')

%  Verticval Relative Error*200:
Z_rel = 200*(Zfilt - LBP(:,3))./ (Zc - LBP(:,3));
[amp,phase,mu,CD,alpha,beta] = sinfit2(Z_rel,PHI(:,2));
%[amp phase]
s = amp*sin(w - phase)+mu;
subplot(212),plot(PHI(:,2),Z_rel,'ow',w,s,'-w',...
        [PHImin PHImax],[mu mu],'-w'),grid
text(PHImin,mu,['mu = ',num2str(mu)]);
text(.75,.4,['CD = ',num2str(CD)],'sc');
text(.75,.37,['amp = ',num2str(amp)],'sc');
text(.75,.34,['phase = ',num2str(phase)],'sc');
title('Vertical'),xlabel('Azimuth, LB (Rad)'),ylabel('Relative Error*200')
pause,clg

%  Elevation Angle:
THETA_rel = (THETA(:,1)-THETA(:,2))./THETA(:,2);
[amp,phase,mu,CD,alpha,beta] = sinfit2(THETA_rel,PHI(:,2));
%[amp phase]
s = amp*sin(w - phase)+mu;
subplot(211),plot(PHI(:,2),THETA_rel,'ow',w,s,'-w',...
        [PHImin PHImax],[mu mu],'-w'),grid
text(PHImin,mu,['mu = ',num2str(mu)]);
text(.75,.9,['CD = ',num2str(CD)],'sc');
text(.75,.87,['amp = ',num2str(amp)],'sc');
text(.75,.84,['phase = ',num2str(phase)],'sc');

title('Elevation Angle'),xlabel('Azimuth, LB (Rad)'),ylabel('Relative Error')


%  LBP:
PHIerr = PHI(:,1)-PHI(:,2);
```

```
[amp,phase,mu,CD,alpha,beta] = sinfit2(PHIerr,PHI(:,2));
%[amp phase]
s = amp*sin(w - phase)+mu;
subplot(212),plot(PHI(:,2),PHIerr,'ow',w,s,'-w',...
      [PHImin PHImax],[mu mu],'—w'),grid
text(PHImin,mu,['mu = ',num2str(mu)]);
text(.75,.4,['CD = ',num2str(CD)],'sc');
text(.75,.37,['amp = ',num2str(amp)],'sc');
text(.75,.34,['phase = ',num2str(phase)],'sc');
title('Azimuth Error'),xlabel('Azimuth, LB (Rad)'),ylabel('Error')
```

PLRPLT1.M generates the polar plots of Figure 10

```
% PLRPLT1.M  plots the Polar Plots of Delta H vs Phi*, and
%          Delta Theta vs Phi*
%    written by Colin R. Cooper          mod.: 5/3/93

clear
fname1=input('Enter the data File Name » ','s');
eval(['load ',fname1]);
fname2 = fname1(1:length(fname1)-4);
% eval(['!del ',fname2,'.met;']);

PHI1 = eval([fname2,'(:,16)']);
PHIlb = eval([fname2,'(:,17)']);
H1 = eval([fname2,'(:,18)']);
Hlb = eval([fname2,'(:,19)']);
THETA1 = eval([fname2,'(:,20)']);
THETAlb = eval([fname2,'(:,21)']);
w = linspace(-pi,pi,100);

% Prepare the plots:
clg
%  Horizontal - DELTA H

Delta_H = 1000*(H1 - Hlb)./Hlb;
[amp,phase,mu,CD,alpha,beta] = sinfit2(Delta_H,PHIlb);
s = amp*sin(w - phase)+mu;

polar(PHIlb,50+Delta_H,'ow',w,50+s,'-w'),grid
%title(['Rel. Horiz. Error * 1000: ',fname1])
% eval(['meta ',fname2]);
pause
Delta_THETA = (THETA1 - THETAlb)./THETAlb;
[amp,phase,mu,CD,alpha,beta] = sinfit2(Delta_THETA,PHIlb);
s = amp*sin(w - phase)+mu;

polar(PHIlb,1+Delta_THETA,'ow',w,1+s,'-w'),grid
%title(['Rel. Elev. Angle Error: ',fname1])
% eval(['meta ',fname2]);
```

APPENDIX D

The KPLOT program is a Graphical Users Interface (GUI) for plotting, examining, and extracting position data of target vehicles at the Nanoose Range. The program is written in the MATLAB programming environment, and even though MATLAB must be available to run the program, it is not necessary that the user be proficient in the MATLAB language.

The GUI format allows the user to use a mouse (or similar pointing device) to select menu options from the graphics screen. To run the program you must enter the MATLAB environment, and at the MATLAB prompt, type:

```
» kplot7
```

The program will load the data sets and generate the main screen (see Figure D-1).

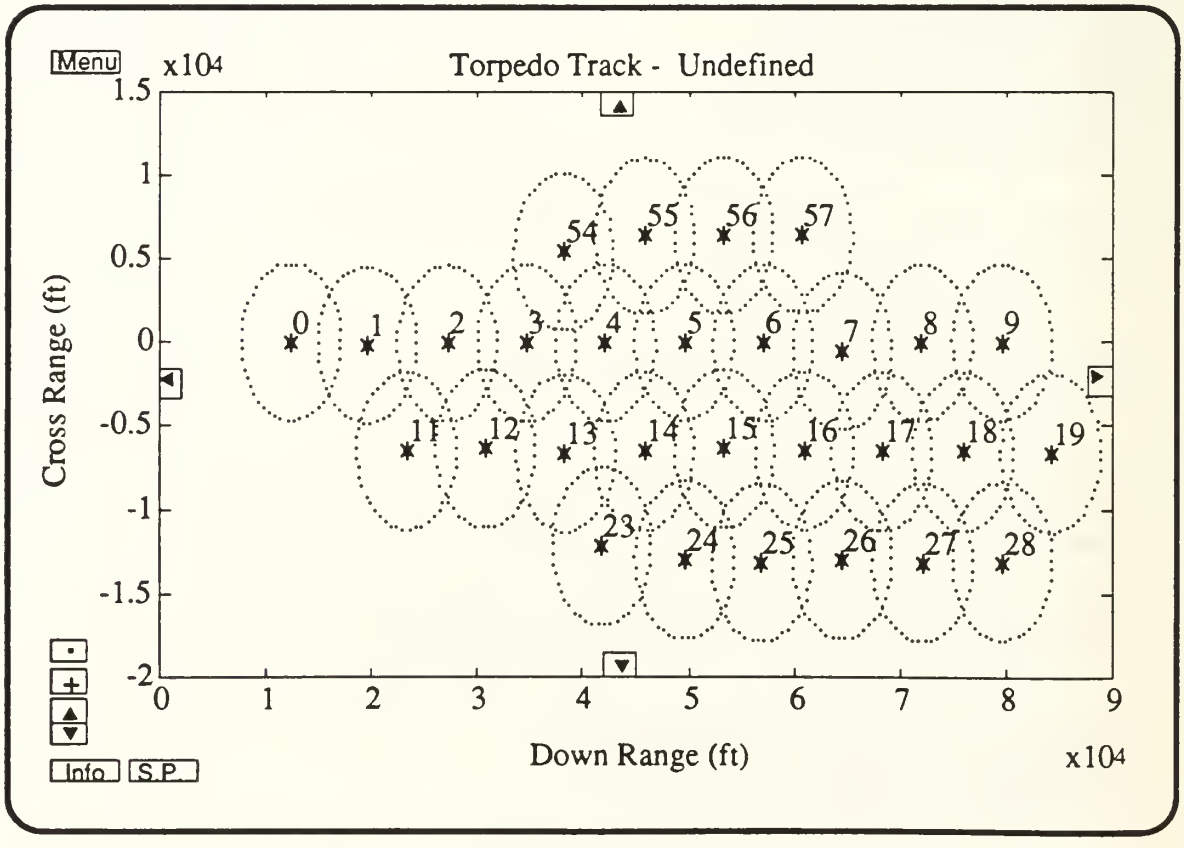


Figure D-1.

In the upper left corner of the screen is a 'menu' button. Selecting this button (clicking once with the mouse) pulls down a set of options for loading data sets, extracting data, creating hard copy of the screen image, and exiting the program (see Figure D-2).

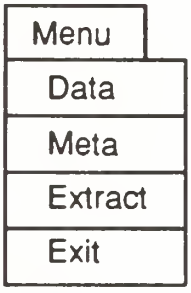


Figure D-2.

Selecting the 'data' option displays a list of available data sets of triple overlap data collected over a three year period (see Figure D-3). A data set may be loaded by entering (at the keyboard) the number of the chosen data set. There will be a brief pause while the data is loaded and sorted, then the program will prompt for another input. The name of the chosen data file should appear at the top of the screen. Once a data file is loaded, entering zero (0) will return to the main screen displaying the data on the Down Range vs. Cross Range plot.

The buttons on the lower left corner of the main screen control the attributes of the screen such as zooming in or out, changing the plotting symbol to '+' or '.', or going into the super plot, 'S.P.', mode of presentation. The zoom-in option,  , allows the user to define a new plot area by clicking the mouse on diagonal corners of the desired area. As the first point is selected it is marked on the screen, and when the second corner is selected the data is replotted on the new axis (see Figure D-4).

TORPEDO TRACK DATA SETS	
Current Data File : Undefined	
1) T2702669.TPL	13) T3011894.TPL
2) T2702670.TPL	14) T3011905.TPL
3) T2702671.TPL	15) T3011906.TPL
4) T2702672.TPL	16) T3011907.TPL
5) T2702673.TPL	17) T3011908.TPL
6) T2702674.TPL	18) T3011909.TPL
7) T3011883.TPL	19) T3011963.TPL
8) T3011884.TPL	20) T3011964.TPL
9) T3011886.TPL	21) T3011965.TP1
10) T3011890.TPL	22) T3011965.TP2
11) T3011892.TPL	23) T3011966.TPL
12) T3011893.TPL	24) T3011967.TPL
	25) T3011968.TPL
0) Return to Main Menu	

Enter your choice »

Figure D-3.

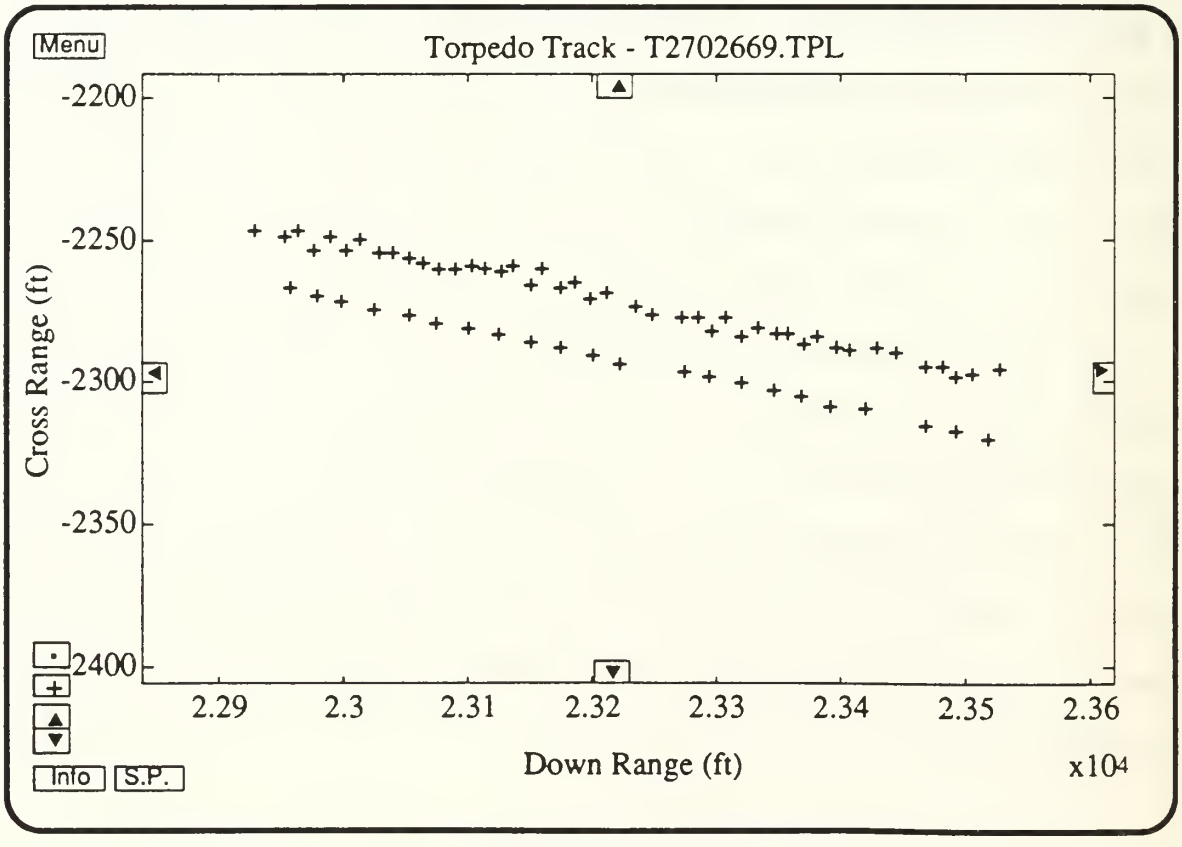


Figure D-4.

The data is initially plotted with a single dot for each data point. When zooming in on a small area of the plot, or if very little data is present, it may be difficult to see the data. The point type buttons, [·] and [+], allow the user to toggle between the two point types to make viewing easier.

The zoom-out button, [▲], may be used with either the left or right mouse button. The left mouse button zooms out with a user specified constant 'zoom factor' in each all directions. The right mouse button takes the screen dimensions back to the original, full range view.

The four arrow buttons on the sides of the plot allow the user to scroll left, right, up and down without changing the dimensions of the view area. Using the left mouse button scrolls with small steps, using the right mouse button scrolls with half screen steps.

The Super-Plot option, [S.P.], captures the data that is currently displayed, and replots it in a new format (see Figure D-5). Each data point is color coded to its contributing array, the point type, '+', 'o', or 'x', represent the depth relative to the other points in the triplet, and a line is drawn from the point towards its contributing array.

Information may be obtained on any point currently displayed. Selecting the info. button, [Info], prompts the user to select a data point on the screen. When the left mouse button is pressed the program will select the nearest data point and list the point count, contributing array, and the x, y, and z coordinates. Pressing any key will return to the graphics screen where another point may be selected. Press the right mouse button to exit the information mode.

Data extraction is performed based on the point count information in the data file. To extract a range of data, first use the info. option to determine the continuous ranges of point counts to extract. Select the extract option from the

menu list, and enter the starting point count, final point count, and the output file name. Multiple point count ranges can be written to the same data file by supplying the same file name each time the data is extracted.

The program may be exited from the main screen by selecting 'exit' from the menu options.

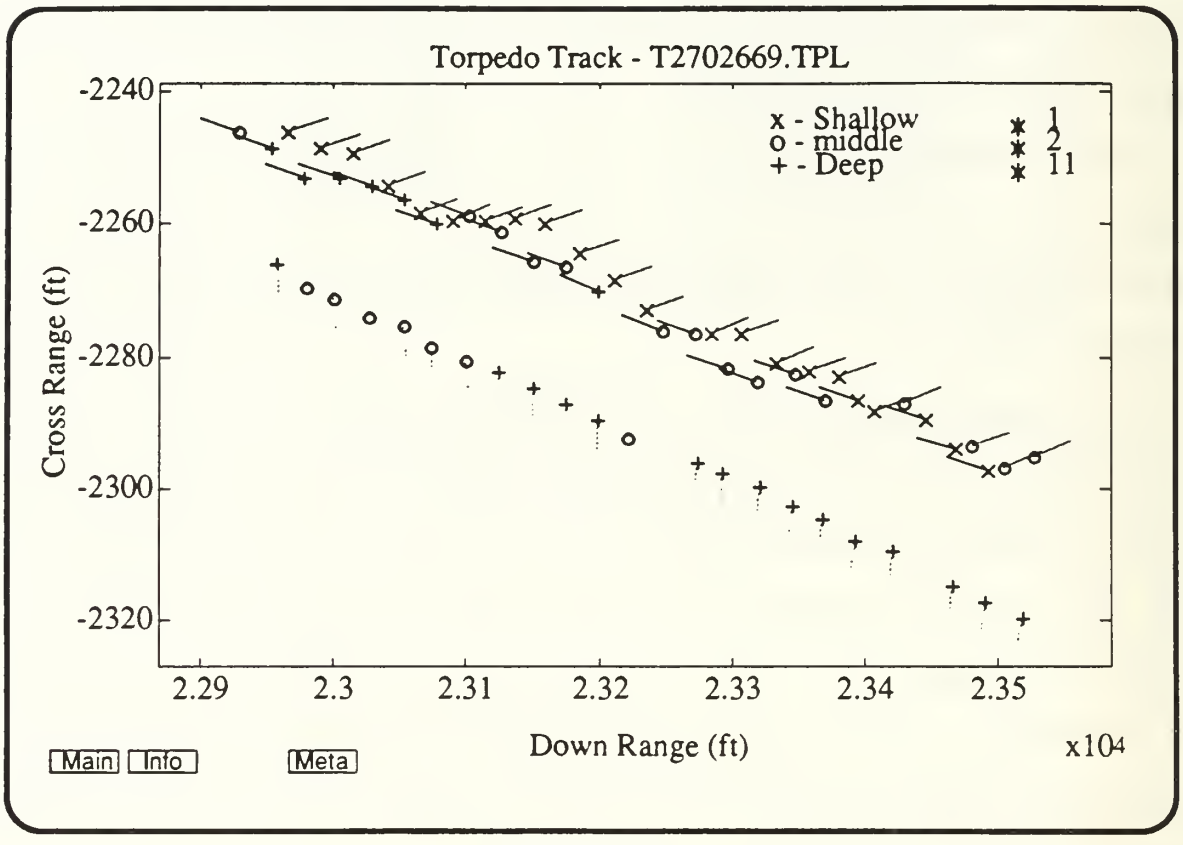


Figure D-5.

MATLAB SOURCE CODE

KPLOT7.M Main driving program for plotting and extraction

```
%     KPLOT7.M is a plotting routine for use with the Keyport
%     Research Project.  It will plot the torpedo tracks and
%     the Accoustic Array locations on the same plot, showing
%     overlap regions and (approximate) array sensitivity zones.

%     This program calls or is supported by:
%           CIR.M, GENARRAY.M, ARRAY.MAT
%           MENUBOX3.M MENUBOX4
```

```
%
%      written by  Colin R. Cooper       Last Mod:  03/04/93

clear;
axis('normal')
paxis = [];                    % Prepare axis vector.
x = []; y = [];                % Prepare data vectors.
pt = '.';                      % Set point plot symbol.
flagm = 0;                     % Flag for making a meta file.
datafile=[' Undefined  '];       % String for data file.
filenames = ['T2702669.TPL'      % Data files available for loading.
          'T2702670.TPL'
          'T2702671.TPL'
          'T2702672.TPL'
          'T2702673.TPL'
          'T2702674.TPL'
          'T3011883.TPL'
          'T3011884.TPL'
          'T3011886.TPL'
          'T3011890.TPL'
          'T3011892.TPL'
          'T3011893.TPL'
          'T3011894.TPL'
          'T3011905.TPL'
          'T3011906.TPL'
          'T3011907.TPL'
          'T3011908.TPL'
          'T3011909.TPL'
          'T3011963.TPL'
          'T3011964.TPL'
          'T3011965.TP1'
          'T3011965.TP2'
          'T3011966.TPL'
          'T3011967.TPL'
          'T3011968.TPL'];

choice = 1;
while choice >= 1;             % Start main menu loop.
clc
  clg                    % Plot the current plot.
  set = [0 1 2 3 4 5 6 7 8 9 11 12 13 14 15 16 17 18 19 ...
         23 24 25 26 27 28 54 55 56 57];    % All arrays.
  if ~isempty(paxis), axis(paxis);
  else, axis([1 2 3 4]);axis;end
  cir(set), hold on
  plot(x,y,[pt,'r'])      % If 'filtered data' doesn't overlap
  title(['Torpedo Track - ' datafile])
  xlabel('Down Range (ft)'),ylabel('Cross Range (ft)')
  paxis = axis; hold off

% CREATE A META FILE IF FLAGM = 1.
  if flagm
```

```
  clc
  flagm = 0;
  flag8 = 1;
  while flag8 > 0       % Loop in case of error.
   disp(' ')
   disp(' ')
   q1 = input('    Enter name of meta file » ','s');  % Input file name.
   if(exist([q1,'.met']) == 2)        % If file exists then ...
    q2 = input('    Append to existing file (y/n) ? » ','s');
    if (q2=='n')|(q2=='N')
     flag8 = 1;              % If 'do not append', go back to top.
    else
     flag8 = 0;              % Else set to write file.
    end
   else
    flag8 = 0;               % File doesn't exist, set to write file.
   end
   if (flag8 == 0)
    disp('              Writing to meta file.')
    eval(['meta ',q1])         % Write to meta file.
   end
  end
end

[choice,bt] = menubox3;

if choice == 1         % ZOOM OUT
  if bt == 1
  clc
  zoom = input('                    Enter Zoom Factor » ');
  rangex = (paxis(2)-paxis(1))*zoom;
  rangey = (paxis(4)-paxis(3))*zoom;
  xloc = paxis(2) - .5*(paxis(2) - paxis(1));
  yloc = paxis(4) - .5*(paxis(4) - paxis(3));
  paxis=[xloc-.5*rangex xloc+.5*rangex yloc-.5*rangey yloc+.5*rangey];
  else
   paxis = [];      % Full Scale
  end

elseif choice == 2     % ZOOM IN
  text(.17,.17,'Select Boundary Points for ZOOM','sc');
  [xloc(1),yloc(1)] = ginput(1);
  polymark(xloc(1),yloc(1),'+c5')
  [xloc(2),yloc(2)] = ginput(1);
  paxis = [min(xloc) max(xloc) min(yloc) max(yloc)];

elseif choice == 3     % SCROLL LEFT
  if bt == 1                    % Scroll by 1/8 Screen
   rangex = (paxis(2)-paxis(1));
   rangey = (paxis(4)-paxis(3));
   xloc = paxis(2) - .5*rangex - .1*rangex;
   paxis=[xloc-.5*rangex xloc+.5*rangex paxis(3) paxis(4)];
```

```
  else                          % Scroll by 1/2 Screen
   rangex = (paxis(2)-paxis(1));
   rangey = (paxis(4)-paxis(3));
   xloc = paxis(1);
   paxis=[xloc-.5*rangex xloc+.5*rangex paxis(3) paxis(4)];
  end

elseif choice == 4     % SCROLL RIGHT
  if bt == 1                    % Scroll by 1/8 Screen
   rangex = (paxis(2)-paxis(1));
   rangey = (paxis(4)-paxis(3));
   xloc = paxis(2) - .5*rangex + .1*rangex;
   paxis=[xloc-.5*rangex xloc+.5*rangex paxis(3) paxis(4)];
  else                          % Scroll by 1/2 Screen
   rangex = (paxis(2)-paxis(1));
   rangey = (paxis(4)-paxis(3));
   xloc = paxis(2);
   paxis=[xloc-.5*rangex xloc+.5*rangex paxis(3) paxis(4)];
  end

elseif choice == 5     % SCROLL UP
  if bt == 1                    % Scroll by 1/8 Screen
   rangex = (paxis(2)-paxis(1));
   rangey = (paxis(4)-paxis(3));
   yloc = paxis(4) - .5*rangey + .1*rangey;
   paxis=[paxis(1) paxis(2) yloc-.5*rangey yloc+.5*rangey];
  else                          % Scroll by 1/2 Screen
   rangex = (paxis(2)-paxis(1));
   rangey = (paxis(4)-paxis(3));
   yloc = paxis(4);
   paxis=[paxis(1) paxis(2) yloc-.5*rangey yloc+.5*rangey];
  end

elseif choice == 6     % SCROLL DOWN
  if bt == 1                    % Scroll by 1/8 Screen
   rangex = (paxis(2)-paxis(1));
   rangey = (paxis(4)-paxis(3));
   yloc = paxis(4) - .5*rangey - .1*rangey;
   paxis=[paxis(1) paxis(2) yloc-.5*rangey yloc+.5*rangey];
  else                          % Scroll by 1/2 Screen
   rangex = (paxis(2)-paxis(1));
   rangey = (paxis(4)-paxis(3));
   yloc = paxis(3);
   paxis=[paxis(1) paxis(2) yloc-.5*rangey yloc+.5*rangey];
  end

elseif choice == 7    % CHANGE POINT PLOT TYPE TO '.'
  pt = '+';

elseif choice == 8    % CHANGE POINT PLOT TYPE TO '+'
  pt = '.';
```

```
elseif choice == 9    % GET INFO ON POINTS.
  choice6 = 1
  clc
  while (choice6 > 0)       % Loop for location of plot points.
    text(.17,.17,'Select A Point (rt to esc)','sc');
    [xloc,yloc,bt] = ginput(1);          % Sample 1 point with mouse.
    if bt == 1
    [xtm,itm] = min(sqrt((x - xloc).^2 + (y - yloc).^2));
    fprintf('\n     Array       = %10.0f\n',ar(itm));
    fprintf('     Point Count = %10.0f\n',pc(itm));
    fprintf('     DownRange   = %10.2f\n',x(itm));
    fprintf('     CrossRange  = %10.2f\n',y(itm));
    fprintf('     Depth       = %10.2f\n\n',z(itm));
    pause
    else
      choice6 = 0;              % Return to main if finished.
    end
  end


elseif choice == 10   % SUPER PLOT.
  exi = find( (x>=paxis(1))&(x<=paxis(2)) & ...
          (y>=paxis(3))&(y<=paxis(4)));  % Index for extracted data.
  pc1 = min(pc(exi)); pc2 = max(pc(exi));
  exi = find((pc>=pc1)&(pc<=pc2));
  kplot6b(pc(exi),x(exi),y(exi),z(exi),ar(exi),paxis,datafile);


elseif choice == 12   % CREATE META FILE
  flagm = 1;


%  OPTION 2 - SELECT TORPEDO TRACK DATA SET
elseif choice == 11
choice2 = 1;
while(choice2 > 0)
st = ['          _       Current Data File :  ' datafile '          _'];
clc
disp(' ')
disp('          ____________________________________________________________ ')
disp('          _            TORPEDO TRACK DATA SETS            _')
disp('          ____________________________________________________________ ')
disp(st)
disp('          ____________________________________________________________')
disp('          _    1) T2702669.TPL   _    13) T3011894.TPL    _')
disp('          _    2) T2702670.TPL   _    14) T3011905.TPL    _')
disp('          _    3) T2702671.TPL   _    15) T3011906.TPL    _')
disp('          _    4) T2702672.TPL   _    16) T3011907.TPL    _')
disp('          _    5) T2702673.TPL   _    17) T3011908.TPL    _')
disp('          _    6) T2702674.TPL   _    18) T3011909.TPL    _')
disp('          _    7) T3011883.TPL   _    19) T3011963.TPL    _')
disp('          _    8) T3011884.TPL   _    20) T3011964.TPL    _')
```

```
disp('          _   * 9)  T3011886.TPL   _    21)  T3011965.TP1    _')
disp('          _    10)  T3011890.TPL   _    22)  T3011965.TP2    _')
disp('          _    11)  T3011892.TPL   _    23)  T3011966.TPL    _')
disp('          _    12)  T3011893.TPL   _   * 24)  T3011967.TPL    _')
disp('          _                        _    25)  T3011968.TPL    _')
disp('          _      * No Data         _                         _')
disp('          ____________________________________________________________________')
disp('          _           0)  Return to Main Menu                  _')
disp('          ____________________________________________________________________ ')
      choice2 = input('                  Enter your choice » ');
      if ((choice2<0)|(choice2>25))|(choice2==24)  % Check for valid choice.
         disp('   !! Not a valid selection !!  Please try again.')
         choice2 = 1;
         pause(3)
      elseif (choice2 == 0)     % Return to main choice, do nothing.


      else
         datafile = filenames(choice2,:);     % Load data file name.
         eval(['load c:\keyport\datafils\',datafile,';']);  % Load data.
         data = eval(datafile(1:8));              % Set variable name.
         pc=data(:,1); x=data(:,2); y=data(:,3);   % Extract data.
         z=data(:,4); ar=data(:,5);
         ind = kpdata(datafile,data(:,1));         % Return 'filtered' indexes.
         xg = data(ind,2); yg = data(ind,3);       % Set 'filtered' values.
         eval(['clear data ',datafile(1:8)]);      % Clear unused variable
      end
   end


%  EXTRACT DATA TO OUTPUT FILE
  elseif choice == 13
    flag8 = 0;
    disp(' ')
    disp(' ')
    disp('   You must define the points by their Point Count Numbers.');
    q1 = input('        Are you ready to proceed ? (y/n) » ','s');
    if (q1 == 'y')|(q1 == 'Y'),
      flag8 = 1;
      disp(' ');
      pc1 = input('    Enter the Starting Point Count » ');
      pc2 = input('    Enter the Final Point Count » ');
      exi = find((pc>=pc1)&(pc<=pc2));     % Index for extracted data.
      clc
    end
    while flag8 > 0       % Loop in case of error.
      disp(' ')
      disp(' ')
      q1 = input('    Enter name of DATA FILE » ','s');  % Input file name.
      if(exist(q1) == 2)        % If file exists then ...
        q2 = input('    Append to existing file (y/n) ? » ','s');
        if (q2=='n')|(q2=='N')
```

```
        flag8 = 1;              % If 'do not append', go back to top.
       else
        flag8 = 0;              % Else set to write file.
       end
      else
       flag8 = 0;               % File doesn't exist, set to write file.
      end
      if (flag8 == 0)            % write to the data file
       disp('             Writing to data file.')
       for i = 1:length(exi)
        fprintf(q1,'%5.0f %10.1f %10.1f',pc(exi(i)),x(exi(i)),y(exi(i)))
        fprintf(q1,' %10.1f %5.0f\n',z(exi(i)),ar(exi(i)));
       end
      end
     end


% KEYBOARD BREAK
 elseif choice == 99
  home
  keyboard              % 'Backdoor' keyboard break (not in menu)

% EXIT THE MAIN PROGRAM
 elseif choice == 14
  disp(' ')
  choice1 = input('Are you sure you want to quit? (y/n) » ','s');
  choice = ((choice1(1)~='y') & (choice1(1)~='Y'));
  clc

 end     % END OF IF LOOPS IN OPTION 5

end
```

MENUBOX2.M Box options on the Super Plot screen.

```
function [c,b] = menubox2
% MENUBOX2.M  draws menu boxes on the plot and returns the
%   option number corresponding to the box chosen using ginput.
%   Called from the KPLOT6B function.

  bx4 = [.01 .07 .07 .01 .01]; by4 = [.01 .01 .04 .04 .01]; % Text box

% MENU BOX
  polyline(bx4,by4,'-w','sc'),text(.015,.01,'Main','sc')
% INFORMATION BUTTON
  polyline(bx4+.07,by4,'-w','sc'),text(.08,.01,'Info','sc')
% META FILE BUTTON
  polyline(bx4+.21,by4,'-w','sc'),text(.222,.01,'Meta','sc')

  c = 0;
```

```
while c == 0;
[x,y,b] = ginput(1,'sc');
 if ((x>=.01)&(x<=.07)) & ((y>=.01)&(y<=.04))
   c = 1;              % Main
 elseif ((x>=.08)&(x<=.14)) & ((y>=.01)&(y<=.04))
   c = 2;              % Info
 elseif ((x>=.22)&(x<=.28)) & ((y>=.01)&(y<=.04))
   c = 3;              % Meta File

 else
   c = 0;
 end
end
```

MENUBOX3.M Main menu for KPLOT7.M returns the option chosen.

```
function [c,b] = menubox3
% MENUBOX.M  draws menu boxes on the plot and returns the
%   option number corresponding to the box chosen using ginput.
%   Called by KPLOT7.M
%
%   written by Colin R. Cooper          Last Mod: 3/4/93

  bx1 = [0 .03 .03 0 0]; by1 = [0 0 .03 .03 0];  % Horizontal
  bx2 = [0 .02 .02 0 0]; by2 = [0 0 .04 .04 0];  % Vertical
  bx3 = [.01 .01 .04 .04 .01 .01 .04];          % Double
  by3 = [.09 .06 .06 .12 .12 .09 .09];          %    box
  bx4 = [.01 .07 .07 .01 .01]; by4 = [.01 .01 .04 .04 .01]; % Text box

% ZOOM IN/OUT BOX
  polyline(bx3,by3,'-w','sc')
  text(.02,.055,'_','sc'), text(.02,.09,'_','sc')
% INFORMATION BUTTON
  polyline(bx4,by4,'-w','sc'),text(.015,.01,'Info','sc')
% SUPER PLOT BUTTON
  polyline(bx4+.07,by4,'-w','sc'),text(.08,.01,'S.P.','sc')
% POINT TYPE BOXES
  polyline(bx1+.01,by1+.13,'-w','sc'),text(.021,.132,'+','sc');
  polyline(bx1+.01,by1+.17,'-w','sc'),text(.021,.175,'.','sc');
% MENU BUTTON
  polyline(bx4,by4+.94,'-w','sc'),text(.015,.95,'Menu','sc')

% SCROLL DOWN BUTTON
  polyline(bx1+.5,by1+.116,'-w','sc'),text(.51,.1155,'_','sc')
% SCROLL UP BUTTON
  polyline(bx1+.5,by1+.908,'-w','sc'),text(.51,.909,'_','sc')
% SCROLL LEFT BUTTON
  polyline(bx2+.112,by2+.52,'-w','sc'),text(.116,.52,'_','sc')
% SCROLL RIGHT BUTTON
  polyline(bx2+.931,by2+.52,'-w','sc'),text(.936,.525,'_','sc')
```

```
% MENU BUTTON

  c = 0;
  while c == 0;
  [x,y,b] = ginput(1,'sc');
  if ((x>=.01)&(x<=.04)) & ((y>=.09)&(y<=.12))
    c = 1;              % Zoom out
  elseif ((x>=.01)&(x<=.04)) & ((y>=.06)&(y<=.09))
    c = 2;              % Zoom in
  elseif ((x>=.112)&(x<=.132)) & ((y>=.52)&(y<=.56))
    c = 3;              % Scroll left
  elseif ((x>=.931)&(x<=.951)) & ((y>=.52)&(y<=.56))
    c = 4;              % Scroll right
  elseif ((x>=.5)&(x<=.53)) & ((y>=.908)&(y<=.938))
    c = 5;              % Scroll up
  elseif ((x>=.5)&(x<=.53)) & ((y>=.116)&(y<=.146))
    c = 6;              % Scroll down
  elseif ((x>=.01)&(x<=.04)) & ((y>=.13)&(y<=.16))
    c = 7;              % Point type '+'
  elseif ((x>=.01)&(x<=.04)) & ((y>=.17)&(y<=.2))
    c = 8;              % Point type '.'
  elseif ((x>=.01)&(x<=.07)) & ((y>=.95)&(y<=.98))    % extended menu
    c=menubox4;
  elseif ((x>=.01)&(x<=.07)) & ((y>=.01)&(y<=.04))
    c = 9;              % Info
  elseif ((x>=.08)&(x<=.14)) & ((y>=.01)&(y<=.04))
    c = 10;              % SuperPlot
  elseif ((x>=.97)&(x<=1)) & ((y>=0)&(y<=.03))
    c = 99;              % Keyboard break
  else
    c = 0;
  end
  end
```

MENUBOX4.M The Extended menu options under the Menu Button.

```
function [c] = menubox4
% MENUBOX4.M  draws menu boxes on the plot and returns the
%   option number corresponding to the box chosen using ginput.
%   Called by MENUBOX3.M
%
%   written by Colin R. Cooper          Last Mod: 3/4/93
  ax = .01; ay = .92;
  bx = [.01 .12 .12 .01 .01 .12 .12 .01 .01 .12 .12];
  by = [.92 .92 .89 .89 .86 .86 .95 .95 .83 .83 .86]; % Text box

% DRAW PULL DOWN MENU
  polyline(bx,by,'-c5','sc')
% DATA FILES
  text(ax+.01,ay,'Data','sc')
```

```
% CREATE META FILE
  text(ax+.01,ay-.03,'Meta','sc')
% EXTRACT DATA FILES
  text(ax+.01,ay-.06,'Extract','sc')
% EXIT THE PROGRAM
  text(ax+.01,ay-.09,'Exit','sc')

 c = 0;
 [x,y] = ginput(1,'sc');
 if ((x>=.01)&(x<=.11)) & ((y>=.92)&(y<=.95))
   c = 11;              % Data files
 elseif ((x>=.01)&(x<=.11)) & ((y>=.89)&(y<=.92))
   c = 12;              % Meta file
 elseif ((x>=.01)&(x<=.11)) & ((y>=.86)&(y<=.89))
   c = 13;              % Extract data
 elseif ((x>=.01)&(x<=.11)) & ((y>=.83)&(y<=.86))
   c = 14;              % Exit program
 else
   c = 15;
 end
```

KPLOT6B.M Super Plot routines

```
function i=kplot6b(pc,x,y,z,a,paxis,datafile)
% kplot6.m is for the advanced plotting routines.
%  Uses color for arrays, and symbols for depth.
%   written by Colin R. Cooper          Last Mod: 3/4/93

clg
rx = max(x)-min(x);  ry = max(y)-min(y);
axis([min(x)-.1*rx max(x)+.1*rx min(y)-.1*ry max(y)+.1*ry]);
arrays = [];
color = [];
cnt = 0;
for i = 1:3:length(pc)
  for j = 1:3
    if ~any(arrays==a(i+j-1))
      arrays = [arrays a(i+j-1)];
      cnt = cnt + 1;
      color = [color cnt];
    end

    arind(j) = find(a(i+j-1)==arrays);    % array index for color.
  end
  [depth,dind]=sort(z(i:i+2));
  plot(x(i+dind(1)-1),y(i+dind(1)-1),['+c',num2str(find(a(i+dind(1)-1)==arrays))],...
       x(i+dind(2)-1),y(i+dind(2)-1),['oc',num2str(find(a(i+dind(2)-1)==arrays))],...
       x(i+dind(3)-1),y(i+dind(3)-1),['xc',num2str(find(a(i+dind(3)-1)==arrays))])
  hold on
end
```

```
[arrays,t] = sort(arrays)

dx = .915;
load array;
xscale = .05*(paxis(2)-paxis(1));
yscale = .75*.05*(paxis(4)-paxis(3));
for i = 1:length(arrays)
  ai = find(num == arrays(i))
  dind = find(a == arrays(i));
  th = atan2((ypos(ai)-y(dind)),(xpos(ai)-x(dind)));
  x2 = xscale*cos(th) + x(dind);
  y2 = yscale*sin(th) + y(dind);
  plot([x(dind)';x2'],[y(dind)';y2'],['-c',num2str(t(i))]);
  polymark(.87,dx,['*c'num2str(t(i))],'sc');
  text(.895,(dx-.015),num2str(arrays(i)),'sc');
  dx = dx - .03;
end

text(.65,.90,'x - Shallow','sc');
text(.65,.87,'o - middle','sc');
text(.65,.84,'+ - Deep','sc');
title(['Torpedo Track - ' datafile])
xlabel('Down Range (ft)'),ylabel('Cross Range (ft)')
hold off
choice1 = 1;
while choice1 > 0
  choice1 = menubox2
  if choice1 == 1
     choice1 = 0;   %return to main program
  elseif choice1 == 2   % GET INFO ON POINTS.
    flag = 1
    clc
    while (flag > 0)        % Loop for location of plot points.
      text(.17,.17,'Select A Point (rt to esc)','sc');
      [xloc,yloc,bt] = ginput(1);          % Sample 1 point with mouse.
      if bt == 1
      [xtm,itm] = min(sqrt((x - xloc).^2 + (y - yloc).^2));
      fprintf('\n     Array       = %10.0f\n',a(itm));
      fprintf('     Point Count = %10.0f\n',pc(itm));
      fprintf('     DownRange   = %10.2f\n',x(itm));
      fprintf('     CrossRange  = %10.2f\n',y(itm));
      fprintf('     Depth       = %10.2f\n\n',z(itm));
      pause
      else
        flag = 0;              % Return to main if finished.
      end
    end     % END INFO LOOP

  elseif choice1 == 3   % CREATE META FILE
    clc
    flag = 1;
    while flag > 0       % Loop in case of error.
```

```
    disp(' ')
    disp(' ')
    q1 = input('   Enter name of meta file » ','s'); % Input file name.
    if(exist([q1,'.met']) == 2)       % If file exists then ...
      q2 = input('   Append to existing file (y/n) ? » ','s');
      if (q2=='n')|(q2=='N')
        flag = 1;              % If 'do not append', go back to top.
      else
        flag = 0;              % Else set to write file.
      end
    else
      flag = 0;                % File doesn't exist, set to write file.
    end
    if (flag == 0)
      disp('              Writing to meta file.')
      eval(['meta ',q1])       % Write to meta file.
    end
  end
 end     % END META LOOP
end      % END WHILE LOOP
```

CIR.M — Overlay of the Nanoose Range Sonar Arrays with areas of sensativity.

```
function cir(a)
%     cir(A) will plot the location and (approximate) circle of
%     sensitivity of each array specified in the vector A.
%     Vector A must be in the following format:
%           A = [1 2 3 4 ... 53]
%     Then enter cir(A).  Available arrays can be looked up in
%     the GENARRAY.M file.

%     by  Colin R. Cooper                    08/24/89
%
%     This file uses data from the Nanoose Range which is stored
%     in a matlab data file array.mat formatted as follows:
%           c    : cosine function for circumference
%           s    : sine function for circumference
%           xpos : x-position of array sensors
%           ypos : y-position of array sensors
%           num  : array numbers
%     To change the size of the circles you must change the
%     c and s vectors in the GENARRAY.M file.  This file generates
%     the current range position of the arrays, and saves the results
%     to the data file ARRAY.MAT.

load array              % Load in data
rx = [];                % Prepare data vectors
ry = [];
cx = [];
```

```
cy = [];
n = length(a);
for k = 1:n
   m = find(num==a(k));     % Locate data for a(k)
   rx = [rx;s+xpos(m)];     % Offset circle by the location of
   ry = [ry;c+ypos(m)];     %   of the array.
   cx = [cx;xpos(m)];       % Locate corresponding array location.
   cy = [cy;ypos(m)];
end
plot(rx,ry,'.w',cx,cy,'*g')  % Plot circles with dots, centers with '*'.

% Now place labels on the arrays
for k = 1:n
   st = int2str(a(k));
   m = find(num==a(k));
   text(xpos(m),ypos(m),st) % Place numbers next to array locations.
end
```

GENARRAY.M creates the data file for the Nanoose Range overlay.

```
%  GENARRAY.M  generates the array.mat file used in the CIR.M
%   plotting routines for KPLOT4.M.
%    Latest information from T3011968.TEX,  1/28/93

d=[12248.3     9.3  -1297.7   0
  19458.8   -174.9  -1308.7   1
  26987.0   -109.8  -1323.2   2
  34500.1    -81.0  -1326.5   3
  42000.8    -55.1  -1318.2   4
  49492.6    -25.2  -1315.5   5
  56927.5    -52.9  -1317.1   6
  64399.1   -576.5  -1349.8   7
  71965.4    -29.2  -1300.8   8
  79473.6    -23.8  -1315.0   9
  75831.8  -6515.0  -1310.2  18
  23169.5  -6488.3  -1312.0  11
  30625.8  -6365.7  -1315.3  12
  38209.3  -6640.7  -1323.0  13
  45642.3  -6512.7  -1327.3  14
  53247.3  -6352.7  -1324.3  15
  60870.9  -6459.2  -1324.3  16
  68213.6  -6524.0  -1313.4  17
  38025.6   5401.9  -1212.6  54
  45698.4   6341.9  -1189.3  55
  53175.8   6417.9  -1218.8  56
  60683.7   6428.5  -1093.3  57
  84053.0  -6595.2  -1302.4  19
  41660.4 -12191.4  -1267.2  23
  49564.5 -12978.0  -1305.7  24
  56822.4 -13154.2  -1204.5  25
```

```
  64438.6  -12971.0  -1255.3   26
  72017.9  -13116.7  -1278.5   27
  79490.1  -13050.2  -1312.2   28 ];
num = d(:,4);
xpos = d(:,1);
ypos = d(:,2);
zpos = d(:,3);
w=linspace(0,2*pi,50);
s=4700*sin(w);
c=4700*cos(w);

% Uncomment the following lines to automatically replace the
%     existing array.mat file.
 save array num xpos ypos zpos s c
 clear d num xpos ypos zpos w s c
```

INITIAL DISTRIBUTION LIST

1. Dudley Knox Library (Code 0142) 1
   Naval Postgraduate School
   Monterey, CA 93943-5002

2. Office of Research Administration (Code 08) 1
   Naval Postgraduate School
   Monterey, CA 93943-5000

3. Department of Operations Research (Code OR) 1
   Naval Postgraduate School
   Monterey, CA 93943-5000

4. Prof. Robert R. Read (Code OR/Re) 3
   Naval Postgraduate School
   Monterey, CA 93943-5000

5. Colin R. Cooper (Code EC/Co) 2
   Naval Postgraduate School
   Monterey, CA 93943-5000

6. Naval Undersea Warfare Engineering Station 2
   Keyport, WA 98345
   ATTN: J. Knudsen, Code 5122